Universidad Carlos III de Madrid
Escuela Politécnica Superior
Grado en Ingeniería Informática
Mención en Ingeniería de Computadores

Exploración del Desarrollo de un Entorno de Internet de las Cosas para la Monitorización de Sensores Ambientales

Trabajo de Fin de Grado

por

Mario Gonzalo Gómez

10 de Octubre de 2017

Tutor:
Álvaro Montero Montes

TRABAJO DE FIN DE GRADO

EXPLORACIÓN DEL DESARROLLO DE UN ENTORNO DE INTERNET DE LAS COSAS PARA LA MONITORIZACIÓN DE SENSORES AMBIENTALES

Autor: **MARIO GONZALO GÓMEZ**
Tutor: **ÁLVARO MONTERO MONTES**

TRIBUNAL

Presidenta: **INÉS MARÍA GALVAN LEÓN**
Secretario: **MARCO ROMANO**
Vocal: **ROCÍO VEGA MARTINEZ**
Suplente: **ISABEL SEGURA BEDMAR**

Tras el acto de defensa y lectura el día 10 de Octubre de 2017 en la **Escuela Politécnica Superior** de la **Universidad Carlos III de Madrid** (Leganés), el tribunal le otorga la siguiente **CALIFICACIÓN**:

Agradecimientos

Me gustaría dedicar este trabajo al tutor del mismo, **Álvaro Montero Montes**, por la inmensa ayuda y apoyo que me ha ofrecido a lo largo del proyecto. No sólo me ha dado ideas y conocimientos para aplicar al proyecto, sino que también me ha motivado continuamente para poder realizarlo. **Muchas gracias por todo, Álvaro.**

De la mismo forma, me gustaría agradecer a **Daniel Jerez Garrido** por estar siempre ahí cuando le necesitaba para cualquier cosa, no sólo durante el proyecto, sino que a lo largo de todos estos años. **Muchas gracias, Dani.**

Por último, a mi familia y amigos, gracias por todo el apoyo, comprensión y aguante que habéis tenido conmigo en todo este tiempo. Cuando peor se ponían las cosas siempre podía contar con vuestro ánimo. **Muchas gracias a todos.**

Dirección
Universidad Carlos III de Madrid
Escuela Politécnica Superior
Avda. de la Universidad
28911 Leganés (Madrid) - SPAIN

Por favor, cita este trabajo como:
Mario Gonzalo Gómez: *Exploración del desarrollo de un Entorno de Internet de las Cosas para la Monitorización de Sensores Ambientales*, TFG, Universidad Carlos III de Madrid, 2017.

Abstract

The main goal of this work is to explore the different configuration to develop an Internet of Things environment using the current tecnology. To know if the developed approaches are useful, this exploration has been framed in the weather data collecting context. To be more precise, this work is focused on developing a weather station that might be distributed in an enclosure. Furthermore, the configuration should allow accessing to data through different devices and modes.

The weather station's main tasks are the collecting of weather data using the sensors, the logging of weather data (if possible) and the visualization of data through a website, Android Application or accessing directly to sensors using a mobile browser.

The system is composed of three main elements: a central computer of data history (this one is a Raspberry Pi), an Android Application and sensors that is formed by a microcontroller NodeMCU connected to an environment sensor such as temperature sensor or humidity sensor. The communication between these all elements is done by Wifi (2.4 Ghz).

Keywords

Internet of Things, Monitoring, Sensors, Exploring, Ubiquitous Computing

Resumen

El objetivo principal de este trabajo es la exploración de diferentes configuraciones para el desarrollo de un entorno que haga uso del concepto Internet of Things aplicando la tecnología actual. Con el objetivo de conocer si los diferentes enfoques desarrollados son de utilidad, se ha decidido enmarcar dicha exploración en el contexto de la extracción de datos medio ambientales. Más concretamente, en el desarrollo de una estación meteorológica que pueda estar distribuida en un recinto. Además de las configuraciones, deberá de permitir el acceso a la información desde diferentes dispositivos o modos.

Las tareas principales de la estación meteorológica son la recopilación de los datos haciendo uso de los sensores del entorno o ecosistema, el registro de los datos (si procede) y la visualización de estos por medio de una página web, una aplicación móvil o directamente accediendo al sensor mediante un navegador móvil.

El sistema se compone de tres elementos principales: un ordenador central alojado en una Raspberry Pi, una aplicación Android y los sensores del sistema que están formados por un microcontrollador NodeMCU conectado a un sensor. Dicho sensor es capaz de leer valores del ambiente como la temperatura y humedad. La comunicación entre todos los componentes del sistema se realiza usando conexiones Wifi dentro del ancho de banda de 2.4 GHz.

Palabras clave

Internet de las Cosas; Monitorización; Sensores; Exploración; Computación Ubicua

Índice general

1. Introducción **15**

1.1. Contexto 15

1.2. Problema 16

1.3. Objetivo 17

1.4. Estructura del documento 18

2. Estado del arte **20**

2.1. Computación ubicua 20

2.2. Internet Of Things 21

2.3. Dispositivos de IoT 23

2.3.1. Raspberry Pi 23

2.3.2. Dispositivos Móviles 25

2.3.3. NodeMCU 26

3. Análisis del sistema **28**

3.1. Requisitos de Usuario 28

3.1.1. Requisitos de Capacidad 30

3.1.2. Requisitos de Restricción 32

3.2. Casos de uso 33

3.2.1. Diagrama general de casos de uso 33

3.2.2. Descripción textual de casos de uso 35

3.2.2.1. Casos de Uso del Escenario de Uso 1 36

3.2.2.2. Casos de Uso del Escenario de Uso 2 37

3.2.2.3. Casos de Uso del Escenario de Uso 3 39

3.3. Requisitos Software 39

3.3.1. Requisitos Funcionales 41

3.3.2. Requisitos No Funcionales 43

3.3.2.1. Operatividad 43

3.3.2.2. Interoperabilidad 44

3.3.2.3. Accesibilidad . . . 45
3.3.2.4. Interfaz . . . 45
3.3.2.5. Usabilidad . . . 46
3.4. Matrices de Trazabilidad . . . 46
3.4.1. Matriz de Trazabilidad Requisitos de Usuario frente Casos de Uso . . . 46
3.4.2. Matriz de Trazabilidad Requisitos de Usuario frente Requisitos Software Funcionales . . . 47
3.4.3. Matriz de Trazabilidad Requisitos Software Funcionales frente Casos de Uso 47

4. Diseño del Entorno 48
4.1. Contexto del Entorno . . . 48
4.2. Arquitectura del Sistema . . . 49
4.3. Componentes de la Aplicación . . . 51
4.3.1. Raspberry Pi . . . 51
4.3.1.1. Servidor . . . 51
4.3.1.1.1. Base de datos . . . 52
4.3.1.1.2. Enrutamiento de direcciones . . . 53
4.3.1.1.3. Cambio de redes . . . 53
4.3.2. Aplicación Android . . . 54
4.3.2.1. Recogida de datos . . . 55
4.3.2.2. Representación de datos . . . 55
4.3.3. Nodos del sistema . . . 56

5. Implementación 57
5.1. Raspberry Pi . . . 57
5.1.1. Aplicación Servidor . . . 57
5.1.1.1. Base de Datos . . . 58
5.1.1.2. Enrutamento de direcciones . . . 59
5.1.1.3. Cambio de redes . . . 60
5.2. Aplicación Android . . . 61
5.2.1. Recogida de datos . . . 62
5.2.2. Representación de datos . . . 62
5.3. Nodos del sistema . . . 63

6. Discusión 64
6.1. Escenarios de Uso del Sistema . . . 64
6.1.1. Raspberry Pi . . . 64

6.1.2. Aplicación Android 65
6.1.3. Nodos del sistema 66
6.1.4. Conclusión Final 66

7. Gestión del Proyecto **67**
7.1. Planificación 67
7.1.1. Ciclo de Desarrollo 68
7.1.2. Planificación Previa 69
7.1.3. Planificación Final 71
7.2. Presupuesto 74
7.2.1. Coste de Personal 74
7.2.2. Coste de Materiales 74
7.2.2.1. Materiales Tangibles 75
7.2.2.2. Materiales No Tangibles 76
7.2.3. Costes Indirectos 76
7.2.4. Coste Total del Proyecto 76

8. Marco Regulador **78**
8.1. Legislación Aplicable sobre la Implementación 78
8.2. Licencias y Tecnologías 78
8.3. Propiedad Intelectual 79

9. Conclusión y Trabajos Futuros **80**
9.1. Conclusión Final 80
9.2. Trabajos Futuros 82

10. Apéndice **83**
10.1. Manual de Usuario 83
10.1.1. Raspberry Pi 84
10.1.1.1. Instalación 84
10.1.1.2. Uso de la Aplicación 84
10.1.2. Aplicación Android 86
10.1.2.1. Instalación 86
10.1.2.2. Uso de la Aplicación 86
10.1.3. Nodos del Sistema 88
10.1.3.1. Instalación 88
10.1.3.2. Uso de la Aplicación 92
10.2. Fallos encontrados 94

10.2.1. Raspberry Pi 94
10.2.2. Aplicación Android 94
10.2.3. Nodos del sistema 94

11. Summary **95**
11.1. Introduction 95
11.1.1. Devices 96
11.1.1.1. Raspberry Pi 96
11.1.1.2. NodeMCU 98
11.2. Objectives 99
11.3. Results 100
11.3.1. Raspberry Pi 100
11.3.1.1. Server Application 101
11.3.1.2. Database 101
11.3.1.3. Address routing 102
11.3.1.4. Networks Switching 103
11.3.2. Android Application 104
11.3.2.1. Data Gathering 104
11.3.2.2. Data Representation 104
11.3.3. System Nodes 105
11.4. Personal Conclusion 106

Índice de figuras

2.1. Internet of Things 22
2.2. Raspberry Pi 3 Model B 24
2.3. NodeMCU ESP8266 26

3.1. Diagrama general Casos de Uso Escenario 1 34
3.2. Diagrama general Casos de Uso Escenario 2 34
3.3. Diagrama general Casos de Uso Escenario 3 35

4.1. Arquitectura general del entorno 49
4.2. Arquitectura del primer escenario de uso 49
4.3. Arquitectura del segundo escenario de uso 50
4.4. Arquitectura del tercer escenario de uso 50

5.1. Ejemplo de registro en MongoDB 59

7.1. Ciclo de desarrollo en cascada retroalimentado 68

10.1. Uso de la Aplicación Interfaz Web Pantalla Principal 85
10.2. Uso de la Aplicación Interfaz Web Pantalla Individual 85
10.3. Uso de la Aplicación Android Pantalla 1 86
10.4. Uso de la Aplicación Android Pantalla 2 87
10.5. Uso de la Aplicación Android Pantalla 3 87
10.6. Uso de la Aplicación Android Pantalla 4 88
10.7. Instalación NodeMCU Componentes 89
10.8. Instalación NodeMCU Paso 1 90
10.9. Instalación NodeMCU Paso 2 90
10.10. Instalación NodeMCU Paso 3 91
10.11. Instalación NodeMCU Paso 4 92
10.12. Uso de Aplicación de Arduino 93

11.1. Raspberry Pi 3 Model B 97

11.2. NodeMCU 98
11.3. General Architecture of the Environment 100

Índice de tablas

3.1. Requisito de Usuario - Ejemplo 29
3.2. Requisito de Usuario - Requisito de Capacidad 01 30
3.3. Requisito de Usuario - Requisito de Capacidad 02 30
3.4. Requisito de Usuario - Requisito de Capacidad 03 30
3.5. Requisito de Usuario - Requisito de Capacidad 04 30
3.6. Requisito de Usuario - Requisito de Capacidad 05 31
3.7. Requisito de Usuario - Requisito de Capacidad 06 31
3.8. Requisito de Usuario - Requisito de Capacidad 07 31
3.9. Requisito de Usuario - Requisito de Capacidad 08 31
3.10. Requisito de Usuario - Requisito de Restricción 01 32
3.11. Requisito de Usuario - Requisito de Restricción 02 32
3.12. Requisito de Usuario - Requisito de Restricción 03 32
3.13. Requisito de Usuario - Requisito de Restricción 04 33
3.14. Caso de Uso - Ejemplo 35
3.15. Caso de Uso CU-E1-01 36
3.16. Caso de Uso CU-E1-02 37
3.17. Caso de Uso CU-E1-03 37
3.18. Caso de Uso CU-E2-01 38
3.19. Caso de Uso CU-E2-02 38
3.20. Caso de Uso CU-E2-03 38
3.21. Caso de Uso CU-E3-01 39
3.22. Caso de Uso CU-E3-02 39
3.23. Requisito de Software - Ejemplo 40
3.24. Requisito Software - Requisito Funcional 01 41
3.25. Requisito Software - Requisito Funcional 02 41
3.26. Requisito Software - Requisito Funcional 03 41
3.27. Requisito Software - Requisito Funcional 04 41
3.28. Requisito Software - Requisito Funcional 05 42

3.29. Requisito Software - Requisito Funcional 06 42
3.30. Requisito Software - Requisito Funcional 07 42
3.31. Requisito Software - Requisito Funcional 08 42
3.32. Requisito Software - Requisito No Funcional 01 43
3.33. Requisito Software - Requisito No Funcional 02 43
3.34. Requisito Software - Requisito No Funcional 03 43
3.35. Requisito Software - Requisito No Funcional 04 43
3.36. Requisito Software - Requisito No Funcional 05 44
3.37. Requisito Software - Requisito No Funcional 06 44
3.38. Requisito Software - Requisito No Funcional 07 44
3.39. Requisito Software - Requisito No Funcional 08 44
3.40. Requisito Software - Requisito No Funcional 09 45
3.41. Requisito Software - Requisito No Funcional 10 45
3.42. Requisito Software - Requisito No Funcional 11 45
3.43. Requisito Software - Requisito No Funcional 12 46
3.44. Matriz de Trazabilidad Requisitos de Usuario frente Casos de Uso 46
3.45. Matriz de Trazabilidad Requisitos de Usuario frente Requisitos Software Funcionales 47
3.46. Matriz de Trazabilidad Requisitos Software Funcionales frente Casos de Uso 47

7.1. Diagrama de Gantt Planificación Inicial 70
7.2. Diagrama de Gantt Planificación Final 72
7.3. Horas totales de trabajo 73
7.4. Salarios de empleados 74
7.5. Coste de materiales físicos 75
7.6. Coste de licencias 76
7.7. Costes indirectos 76
7.8. Costes materiales 76
7.9. Coste total del proyecto 77

Capítulo 1

Introducción

El primer capítulo de este documento tiene como propósito introducir al lector en el marco en que se mueve este trabajo además de exponer los objetivos planteados a realizar. Para ello, en la primera sección se explica el contexto o área donde está enmarcado este trabajo. Posteriomente, se hará mención de algunos de los problemas existentes en este campo para delimitar en la siguiente sección los objetivos que se intentarán alcanzar al finalizar este proyecto. Por último, se presenta la estructura del documento indicándose los capítulos que lo formarán junto con una descripción detallada del contenido que será tratado en cada uno de ellos.

1.1. Contexto

En los dos últimos siglos, la sociedad ha podido ser testigo de cómo la tecnología ha ido cambiando la vida cotidiana de las personas. Entre los hitos más destacables en el ámbito de las computación y comunicaciones, cabe destacar la aparición de los ordenadores personales y el acceso a internet. Estos dos elementos permitieron que una gran cantidad de organizaciones, empresas y algunas personas (con acceso a estos avances) pudieran intercambiar una gran cantidad de información de punto a punto en un tiempo razonable. Pero de cualquier manera, el cambio más drástico y radical que la sociedad continúa viviendo es la proliferación de los dispositivos móviles inteligentes. La mayoría de las personas ha pasado de mandar cartas o hacer llamadas desde un teléfono de red fija a hacer estas tareas y muchísimas más en cualquier lugar o momento utilizando sus teléfonos móviles. La posibilidad de compartir información o chatear de manera instantánea, incluso con personas que se encuentran en la otra punta del mundo, ha permitido eliminar por completo las barreras que impedían el acceso a la comunicación e información.

Esta época de revolución tecnológica permite el acceso a la información y la comunicación a gran parte de la población que se encuentra en los países desarrollados o en vías de desarrollo. Hoy día, el volumen de información que se sube y descarga en internet es casi inimaginable. Hasta hace dos décadas, la información que se encontraba en internet tenía que ser introducida por personas que tenían los conocimientos necesarios para ello. Sin embargo, gracias al desarrollo de la Web 2.0 o Web Social [6], las facilidades para que cualquier usuario pueda compartir contenido como vídeos, conocimiento, opiniones o comentarios se han visto incrementadas notablemente. Lo que ha dado lugar a que en internet más personas puedan contribuir con su información, lo cual ha llevado de la misma forma a un crecimiento en la cantidad de información total que está "colgada" en Internet. Aunque parezca difícil, esta tendencia de crecimiento no sólo continuará, sino que se verá potenciada por el incremento del número de dispositivos "inteligentes" que se conectarán a internet como lavadoras, neveras, alarmas, cámaras de tráfico, etcétera. En la última década, varios trabajos han intentado estimar el número de dispositivos que estarán conectados a Internet para el 2020 [7]. La cantidad total puede cambiar por la evolución final pero no estará lejos de los 25.000 millontes. En cualquier caso, la conclusión es que será mucho mayor que la actual, que se sitúa en los 8300 millones de dispositivos (más de uno por cada ser humano de la tierra) [2].

La utilidad de toda la información que proporcionan estos datos actualmente, sirve de gran utilidad para tareas como: realizar la lista de la compra, comprobar que la calefacción esté encendida minutos antes de llegar a casa o que los despertadores suenen antes si hay tráfico denso o un accidente en el trayecto al trabajo. Es imposible no afirmar que en los próximos años su uso se podrá extender a muchos más campos, este trabajo se centrará la exploración del desarrollo de esta tecnología en el marco de una estación meteorológica distribuida. Este sistema podría ser útil para la monitorización de cultivos que requieran un cuidado adicional, para alcanzar una mejor calidad como viñedos o como sistema para monitorizar las características de una región con el objetivo de construir aeropuertos (para el estudio del viento) o carreteras (para evitar zonas de nieblas casi permanentes [1]).

1.2. Problema

Hoy día, los ordenadores y dispositivos móviles como teléfonos móviles están diseñados y preparados para ayudar a sus usuarios a resolver problemas como encontrar la solución de una ecuación compleja, pero también cotidianos como encontrar el restaurante más cercano, planificar la ruta más rápida hasta él o incluso permitirnos pagar la cuenta. De cualquier manera, es cierto que aunque los ordenadores parezcan inteligentes no son más que máquinas que siguen una codificación, por lo que los encargados de dotar una funcionalidad útil somos los seres humanos.

Por tanto, tenemos que orientar esta tecnología de tal forma que ayude en tareas de la vida cotidiana a un gran número de personas. Por ejemplo, sería muy interesante dotar a las casas de la gente con un ordenador central que proporcionara información a los residentes. Este ordenador tendría acceso a la red de internet y podría avisar del pronóstico del tiempo que va a hacer a lo largo del día antes de ir a trabajar, e incluso si deberíamos llevar paraguas. El sistema podría indicarnos antes de salir a trabajar de que hay un atasco en nuestra ruta habitual y ofrecernos una ruta alternativa, o que activara un despertador con mayor antelación para evitar el atasco si se produce de manera habitual. De manera doméstica, el sistema podría estar conectado a nuestra televisión y sugerir películas o series parecidas a las ya vistas, podría tener un inventario de los productos en la nevera y mostrarlos en una aplicación de móvil o incluso sistemas como el de la calefacción o la luz podrían ser controlados de manera remota.

El abanico de posibilidades es muy amplio, sin embargo existen muchos problemas de compatibilidad y el concepto todavía no está muy asimilado por la sociedad. Los problemas de compatibilidad radican principalmente en el software de los dispositivos, que está hecho de manera específica para cada componente. Este hecho provoca que los sistemas centralizados que hemos comentado anteriormente estén presentes en casas muy modernas y con este propósito en concreto. Por tanto, esta tecnología que debería distribuirse ampliamente por la sociedad queda reducida a un grupo selecto, debido precisamente a la exclusividad de su software. Las personas por lo general no quieren aprender a manejar una gran variedad de programas o llevar instaladas muchas aplicaciones en el móvil para controlar una serie de aparatos de uso “trivial”, por lo que sería ideal agrupar todas estas funcionalidades.

En este proyecto vamos a intentar dar solución a este incoveniente que se presenta en la actualidad creando un sistema accesible desde varios puntos. El hecho de ofrecer múltiples vías de acceso permite que un mayor número de usuarios tenga la posibilidad de conocer esta información.

1.3. Objetivo

La motivación de este proyecto es la de crear un entorno ubicuo que esté en todo momento comunicado entre sí y pueda ofrecer la información recopilada al usuario de una manera sencilla. Este entorno ubicuo va a tener una temática de una estación meteorológica que interpreta la información del área en que se encuentra y la transmite a los usuarios. Se van a desarrollar tres maneras de acceder a este contenido, dos de las cuales estarán centralizadas en un servidor y una tercera que es totalmente descentralizada.

Entre los dos métodos que forman un sistema centralizado se va a encontrar una página web y una aplicación Android que permita el acceso a la información. Ambos métodos van a permitir la lectura de un registro histórico de los datos. La forma de acceso no centralizada va a permitir a los usuarios conectarse directamente a los nodos del sistema para leer su información en tiempo real.

1.4. Estructura del documento

Obviando el presente capítulo, los sucesivos capítulos que constituyen este documento son los siguientes:

- **Estado del arte:** esta sección tiene como objetivo explicar el estado actual de los conceptos sobre los que versa este proyecto, que son la Computación Ubicua y el Internet de las Cosas (Internet of Things). También se presentará en detalle algunos dipositivos que de alguna manera están relacionados con los anteriores conceptos y han sido usados para este trabajo.

- **Análisis:** en este apartado se presenta el estudio software inicial que tuvo que ser realizado para la implementación de este proyecto. Para ello, a lo largo del capítulo se mostrarán los requisitos que fueron extraídos o impuestos por el tutor, que realizó el rol de cliente, los casos de uso que se pudieron identificar y los requisitos software que describen el proyecto en detalle para su realización por programadores.

- **Diseño:** este apartado tiene como objeto presentar el diseño que satisface los requisitos impuestos en el análisis del sistema. Para ello, se hará uso de diagramas que presentan la arquitectura software que será seguida en cada uno de los escenarios/configuraciones que se van a implementar.

- **Implementación:** este apartado expondrá a muy bajo nivel las decisiones que se tuvieron en cuenta cuando se realizó la codificación del código siguiendos los diseño planteados en el apartado anterior.

- **Discusión:** este apartado se centrará en analizar la exploración realizada en este proyecto. Para ello se enfocará en estudiar las configuraciones desarrolladas en cuanto a sus ventajas y limitaciones y los problemas que se han tenido durante el desarrollo de este proyecto para mostrar a qué dificultades se tiene que enfrentar un desarrollador.

- **Gestión:** esta sección recopila los datos de planificación y presupuestos totales que ha tenido el proyecto. Dentro de la planificación se presenta una comparación entre la planificación inicial

y la planificación final resultante. Los presupuestos abarcan los costes totales de llevar a cabo este proyecto.

- **Marco regulador:** este punto tiene como objetivo explicar los aspectos legales referentes al proyecto.
- **Conclusiones y Trabajos Futuros:** en esta sección describe todos los resultados que se pueden extraer tras la finalización del proyecto. En este apartado se detallarán los objetivos alcanzados y las limitaciones y problemas encontrados tras su desarrollo. También se analizan trabajos futuros que se podrían llevar a cabo para resolver o mitigar las limitaciones detectadas. Además de unas conclusiones personales sobre la realización de este trabajo fin de grado.
- **Apéndice:** este apartado contiene un manual de usuario para la instalación y manejo de las aplicaciones, y una serie de fallos que se pueden dar en las aplicaciones del sistema.

Capítulo 2

Estado del arte

Antes de presentar el análisis y diseño del proyecto desarrollado, se van a introducir una serie de conceptos que forman parte de este trabajo. Lo primero que se va a explicar es el término computación ubicua para describir a continuación el término Internet of Things que está muy relacionado. Finalmente, se va a presentar una serie de dispositivos que fueron diseñados para abarcar estos conceptos y que han sido empleados en este proyecto.

2.1. Computación ubicua

El concepto de computación ubicua hace referencia a la presencia de ordenadores en objetos que no aparentan ser computadoras a primera vista. A diferencia de los ordenadores tradicionales, la computación ubicua se puede dar en cualquier dispositivo, en cualquier lugar y utilizando cualquier tipo de formato de datos. Estos ordenadores se encuentran normalmente embebidos o empotrados dentro de objetos de la vida cotidiana de las personas, para reducir la necesidad de interactuar con ordenadores de manera tradicional y hacerlo de una forma más natural, prácticamente intentado que el usuario no se de cuenta de que está manejando un ordenador.

Este concepto es introducido en el año 1988 por Mark Weiser mientras trabajaba en el Centro de Investigación Xerox de Palo Alto (PARC). Tres años más tarde escribiría el artículo *"The Computer for the Twenty-First Century"* [12], que provocó la difusión mundial de este concepto.

Los principales objetivos que defiende el concepto de computación ubicua son:

- **Invisibilidad.** El autor defiende que las tecnologías que más impacto tienen son aquellas que "desaparecen" con el tiempo. Son aquellas a las cuales nos acostumbramos tanto que se convierten en elementos cotidianos de nuestras vidas. La caracterísitica principal de este tipo de computación es que utiliza elementos de pequeño tamaño que pueden ser embebidos dentro de objetos, lo cual hace que permanezcan ocultos.

- **Uso eficiente de espacios inteligentes.** El hecho de poder asociar estos ordenadores con sensores y actuadores permiten una interacción de las personas u otros dispositivos con los elementos físicos del entorno, utilizando esta tecnología como intermediaria.

- **Escalabilidad.** Uno de los principales fines de esta tecnología es expandirse y crear una mayor red de elementos comunicándose entre sí.

- **Enmascarar desigualdades tecnológicas del ambiente.** El desarrollo tecnológico disponible en un ambiente puede ser más avanzado que en otros, lo cual porovocaría en el usuario una sensación de desnivel entre diferentes entornos. Esto es algo que no se da hoy en día debido al aislamiento que sufren estos entornos entre sí.

El autor opina que los ordenadores tradicionales de sobremesa van a ser sustituidos por este tipo de computadores invisibles y distribuidas por el entorno, que van a integrarse en la vida cotidiana de las personas de forma natural, como si de ropa se tratase. Esta idea que definía a principios de los años 90 se ha convertido en una realidad hoy en día, pues dispositivos como el teléfono móvil se han convertido en piezas esenciales en la vida de las personas.

2.2. Internet Of Things

El Internet de las Cosas o Internet of Things (IoT) es un concepto que evoluciona directamente de la computación ubicua, aunque su origen puede ser anterior. Esta idea representa la conexión entre múltiples dispositivos y elementos de la vida cotidiana de las personas entre sí utilizando internet, formando una gran red de información. Cada uno de estos nodos que componen la red tiene la capacidad de interactuar con personas o con otros dispositivos para ofrecer información. El objetivo de este concepto es llegar a un punto en el que la mayoría de los aparatos que utilizamos en nuestro día a día puedan ser accedidos o controlados a través de Internet.

Figura 2.1: Internet of Things

Fuente: http://blog.conento.com/es/iot-bigdata/

El primer dispositivo diseñado con esta idea fue una máquina de refrescos en el año 1982, anterior al concepto de computación ubicua acuñado por Mark Weiser. Esta máquina de refrescos estaba conectada a la red y era capaz de transmitir datos como el inventario actual. En el año 1990 se crea una tostadora que es capaz de ser encendida y apagada a través de peticiones TCP/IP desde un ordenador.

El concepto de *"Internet of Things"* fue introducido por primera vez en 1999 por Kevin Ashton mientras investigaba la idea de introducir identificadores de radio frecuencia (RFID) en su proyecto. Utilizó el concepto de "Internet" con la única intención de atraer la atención de los ejecutivos de su empresa, porque en esa época estaba muy de moda esta nueva tecnología. Sin embargo, este concepto cayó en el olvido y no volvió a aparecer hasta 10 años después.

El crecimiento de esta idea se produce alrededor de los años 2008 y 2009, donde el número de dispositivos portátiles conectados a la red incrementó su número de manera drástica. Los dispositivos más significativos que surgieron en estos años fueron los teléfonos inteligentes o smartphones, cuya difusión por la población tuvo un ritmo frenético. La existencia de un mayor número de redes wifi de acceso público tuvo una gran repercusión en ese crecimiento.

Las principal aplicación de este concepto es crear una gran red de dispositivos conectados que comparten información. Este tipo de información puede ser de múltiples tipos. Por ejemplo, desde un dispositivo móvil se puede acceder al estado de tráfico de una red de carreteras en tiempo real; también se puede utilizar esta tecnología para ver contenido multimedia como pueda ser un streaming en directo mientras caminamos por la calle.

Otro tipo de objetivos que puede llevar a cabo esta tecnología es el aprendizaje en el ámbito educativo. Estudios como [5] han hecho un análisis de cómo la introducción del IoT en clases universitarias ha intervenido en el proceso de aprendizaje.

2.3. Dispositivos de IoT

Existe una gran variedad de dispositivos y elementos que pertenecen al Internet of Things, pero principalmente se agrupan en: ordenadores, microcontroladores, sensores y actuadores. Los ordenadores que más se están utilizando en este tipo de entornos son los dispositivos móviles como los smartphones y tablets. El gran número de unidades con las que cuenta este tipo de dispositivos ha sobrepasado a los tradicionales ordenadores de sobremesa. Los microcontroladores son ordenadores cuya funcionalidad queda reducida a una serie de tareas limitadas, y se encuentran principalmente asociados a sensores y actuadores para interactuar con el entorno. Por lo general, los micrcontroladores están empotrados en otros elementos del sistema y no son visibles.

2.3.1. Raspberry Pi

La Raspberry Pi es un ordenador que se estructura en una única placa de dimensiones reducidas y con un coste bajo. Está orientada principalmente a la enseñanza de múltiples ámbitos de la informática y a la creación de proyectos caseros o “DIY” (Hazlo tú mismo).

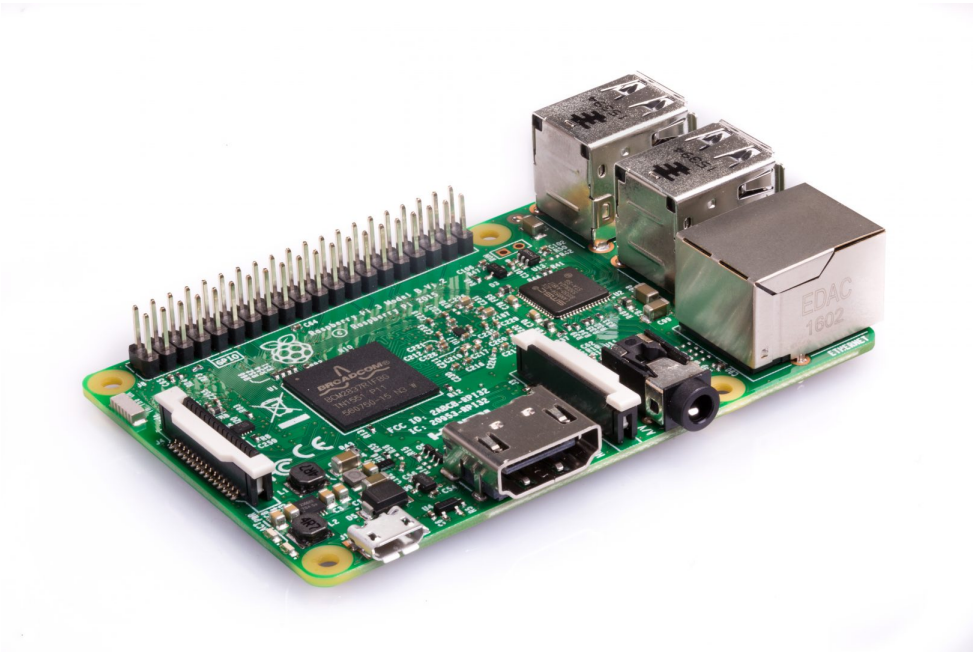

Figura 2.2: Raspberry Pi 3 Model B

La placa cuenta con una arquitectura ARM sobre la cual monta el sistema operativo oficial Raspbian, que es una versión de Debian. Se pueden instalar otros sistemas operativos basados en Linux e incluso algunos otros como Windows 10 IoT.

Como cualquier otro ordenador, la Raspberry Pi se compone de una serie de puertos USB, HDMI, Ethernet, toma de audio jack, tarjeta de memoria SD y módulo wifi en su última versión. La placa monta un procesador ARM 64-bits de cuatro núcleos a 1.2 GHz cada uno con 1 GB de memoria RAM, lo cual permite un nivel alto de cómputo. Además cuenta con 40 pines de GPIO y conectores preparados para conectar un display y una cámara directamente a la placa.

La principal característica que convierte este ordenador en un producto tan interesante, aparte de su tamaño comparable a una tarjeta de crédito, es su bajo precio. Tiene un coste aproximado de unos 30 euros y hace que sea accesible para un amplio conjunto de usuarios.

Está destinada principalmente a la educación y aprendizaje de usuarios iniciados en la programación y manejo de ordenadores. Es por ello que el sistema operativo oficial Raspbian incluye un gran número de aplicaciones de aprendizaje centradas en la programación como Scratch [11] .

El abanico de posibilidades que ofrece la Raspberry Pi a la hora de crear proyectos sólo está limitado por nuestra propia imaginación. Algunos de los ejemplos que se pueden encontrar son: reconocimiento facial [8], control remoto de robots [10], monitorización y transmisión de datos [3] e

incluso sistemas de conducción autónoma [9].

Una de las aplicaciones más llamativas de la combinación de la Raspberry Pi con micrcontroladores como Arduino o NodeMCU es la de dotar de inteligencia al hogar. Libros como [4] aportan ideas y explicaciones de como llevar a cabo proyectos caseros para este ámbito.

2.3.2. Dispositivos Móviles

La verdadera expansión de la tecnología del Internet of Things ha sido gracias al desarrollo de los teléfonos inteligentes y las tablets. Estos dispositivos portátiles tienen una capacidad computacional ligeramente menor a la de los ordenadores de sobremesa, pero es más que suficiente para este tipo de entornos.

A pesar de que esta tecnología ha tenido su gran expansión en los últimos 10 años, su concepción es bastante anterior. En la década de los 70 ya se estaban explorando las posibilidades de incorporar el concepto de telefonía a los ordenadores. Fue en el año 1992 cuando IBM creó un prototipo de un teléfono móvil que incorporaba características de PDA (Asistente personal digital). Dos años despúes, el primer smartphone fue inventado y llamado Simon Personal Communicator. Este dispositivo creado por BellSouth era capaz de realizar y recibir llamadas, enviar emails y otras aplicaciones. A finales de la década de los 90 se hizo común el uso de PDAs, algunas de las cuales ya empezaba a introducir conectividada a internet.

Sin embargo, la gran revolución de los teléfonos inteligentes llegaría en el año 2007. A principios de este año, la innovadora empresa Apple lanza al mercado el que se ha convertido en el mayor icono de este tipo de dispositivos, el primer iPhone. A finales de este mismo año, Google anuncia su competencia con el nuevo sistema operativo de uso libre para este tipo de sistemas, Android. Microsoft también estaba representada en esta tecnología con el sistema operativo Windows Mobile, que posteriormente no tuvo tanto éxito como estos dos primeros.

Hasta esta fecha, mucha gente disponía de un teléfono móvil para comunicarse. Sin embargo, a partir de ese año, esta tecnología se convirtió en un elemento casi de necesidad diaria para los usuarios. La introducción de aplicaciones como Whatsapp o Telegram, que utilizan internet como medio de comunicación, cambió por completo el formato de comunicación entre usuarios, sustituyendo a la tradicional mensajería SMS y reduciendo el número de llamadas realizadas.

Los smartphones más modernos pueden contar con más de 10 sensores integrados, capaces de realizar tareas como ubicar la localización del dispositivo o evaluar condiciones ambientales. Todos ellos juntos pueden llegar a recopilar una gran cantidad de datos. Si a ello le sumamos elementos como las pulseras o relojes inteligentes, el smartphone adquiere un nuevo papel en nuestras vidas, el de un cerebro digital capaz de almacenar y comunicar este tipo de datos. Tecnologías como el NFC (Near Field Communications) presentes en los dispositivos móviles permiten controlar otros dispositivos de manera remota.

Todas estas aplicaciones se relacionan directamente con el concepto de Internet of Things, y sólo queda ver en qué puede evolucionar todo esto en un futuro cercano.

2.3.3. NodeMCU

La placa NodeMCU es un microcontrolador capaz de interactuar con distintos elementos electrónicos haciendo uso de un lenguaje de programación interpretado basado en Lua y de código libre . Al igual que la Raspberry Pi, la principal motivación de este tipo de placas es la de aprendizaje y creación de proyectos caseros.

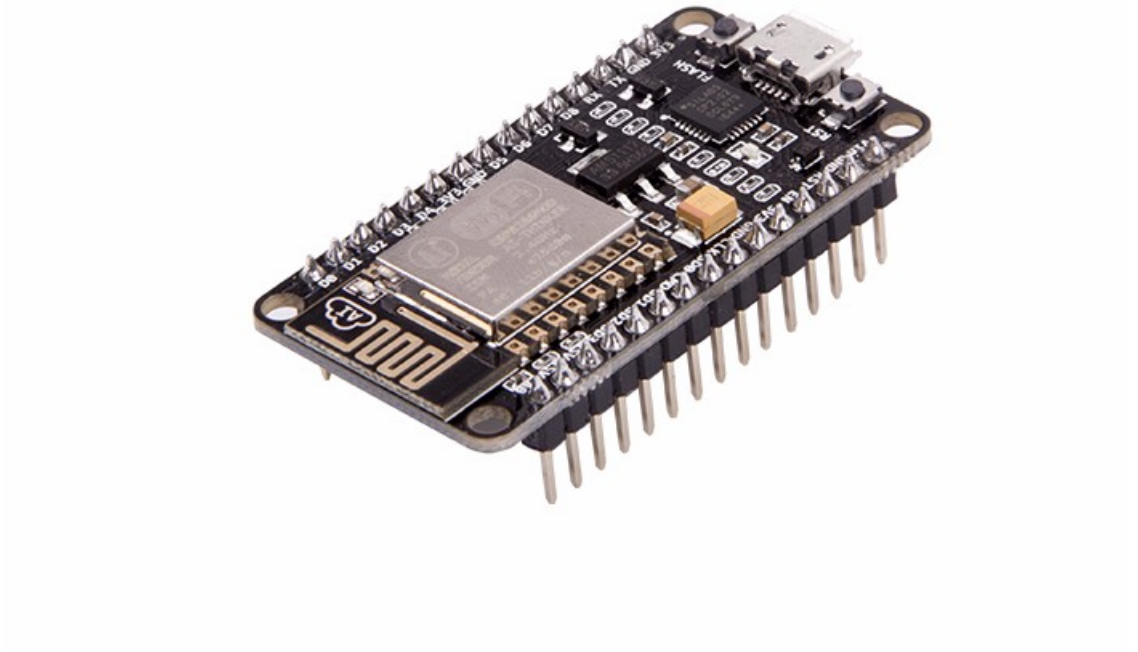

Figura 2.3: NodeMCU ESP8266

Un microcontrolador se puede definir como un ordenador que no dispone de un sistema operativo convencial y tiene un conjunto de tareas a realizar reducido. Los microcontroladores suelen ser el enlace entre los dispositivos electrónicos como sensores o actuadores y ordenadores que actúen como controladores o maestros del sistema.

Los microcontroladores usados para los proyectos *DIY* más comunes son las placas Arduino. Existen diversas versiones de esta placa, siendo la más común la Arduino Uno. La arquitectura general de estas placas se compone de: conector USB, conector de alimentación, una serie de pines digitales (algunos de ellos simulan ser analógicos), pines de alimentación y botón de reset. Su componente principal es es el microcontrolador ATmega que tiene distintas versiones. Modelos más modernos como el que se utiliza para este proyecto llevan incorporados módulos WiFi que permiten su conexión a internet, ampliando las posibilidades de proyecto al Internet of Things (IoT).

La placa utilizada en el proyecto se llama NodeMCU y dispone de un módulo wifi ESP8266, lo cual habilita la comunicación inalámbrica mediante internet. Esta característica permite alojar un servidor web en el microcontrolador para que, además de leer datos o activar sensores en el sistema, pueda representar estos contenidos en su propia página web.

De la misma forma que las placas Arduino, la NodeMCU se puede programar en el lenguaje de programación Arduino utilizando su entorno de desarrollo para ello. Este lenguaje de programación es una variante del tradicional lenguaje C y que, al ser de software libre, permite que la comunidad de usuarios desarrolle librerías externas.

Capítulo 3

Análisis del sistema

El proceso previo al desarrollo del proyecto es el de análisis, en el que se evalúan los requisitos impuestos y los flujos de funcionamiento del sistema. Esta tarea se lleva a cabo simulando un proyecto real en el que el cliente impone las restricciones del sistema y el jefe de proyecto las interpreta en forma de requisitos y casos de uso. Para esta simulación el cliente ha sido el tutor del proyecto y el jefe de proyecto ha sido el autor del trabajo.

Dentro de los requisitos del sistema se tienen que diferenciar los requisitos de usuario y los de software. Junto con los requisitos de usuario se representan los distintos casos de uso que definen la acciones que un usuario puede realizar sobre el sistema. De estos requisitos y casos de uso derivan los requisitos de software.

3.1. Requisitos de Usuario

Los requisitos de usuario representan todas las funcionalidades y restricciones impuestas en el sistema tanto por el cliente como por el jefe de proyecto en una primera aproximación. Estos requisitos se dividen en:

- **Requisitos de capacidad:** definen las funcionalidades de las que va a disponer el sistema.
- **Requisitos de restricción:** indica las limitaciones que se imponen al sistema así como las propiedades de las capacidades de este.

Cada uno de los requisitos de usuario se van a mostrar en una tabla con la siguiente estructura 3.1:

RUC-EX-XX	
Nombre	
Descripción	
Fuente	
Necesidad	
Prioridad	
Pre-requisito	

Tabla 3.1: Requisito de Usuario - Ejemplo

Los campos contenidos en la tabla se describen a cotinuación:

- **Identificador:** este campo sirve para identificar inequívocamente cada uno de los requisitos desarrollados. La estructura general del identificador sigue la siguiente forma: RUC-XX. Está compuesto por dos partes: la primera indica que se trata de un requisito de usuario (RU) y la segunda determina el número que identifica al requisito. Al prefijo se le añade una "C" si es un requisito de capacidad y una "R" si es un requisito de restricción.

- **Nombre:** nombre asociado con el requisito. Sirve para dar una idea general del objetivo del mismo.

- **Descripción:** explicación detallada del objetivo del requisito.

- **Fuente:** indica qué actor ha propuesto el requisito en cuestión. Los actores pueden ser: cliente o jefe de proyecto.

- **Necesidad:** representa el grado de urgencia de llevar a cabo el requisito. Los posibles valores son: opcional, deseable y esencial.

- **Prioridad:** indica el nivel de urgencia de que esté presente el requisito en el sistema. Los valores posibles son: baja, media y alta.

- **Pre-requisito:** en esta sección se indican otros requisitos que son necesarios para llevar a cabo el requisito en cuestión.

3.1.1. Requisitos de Capacidad

RUC-01	
Nombre	Valor de la temperatura en tiempo real.
Descripción	El sistema tendrá la capacidad de dar en tiempo real el valor de la temperatura ambiente del entorno.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.2: Requisito de Usuario - Requisito de Capacidad 01

RUC-02	
Nombre	Valor de la humedad en tiempo real.
Descripción	El sistema tendrá la capacidad de dar en tiempo real el valor de la humedad del entorno.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.3: Requisito de Usuario - Requisito de Capacidad 02

RUC-03	
Nombre	Valor histórico de la temperatura.
Descripción	El sistema permitirá obtener un registro histórico de la temperatura ambiente en el entorno en los siete últimos días.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.4: Requisito de Usuario - Requisito de Capacidad 03

RUC-04	
Nombre	Valor histórico de la humedad.
Descripción	El sistema permitirá obtener un registro histórico de la humedad en el entorno en los siete últimos días.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.5: Requisito de Usuario - Requisito de Capacidad 04

RUC-05	
Nombre	Interfaz web.
Descripción	Los datos del sistema podrán ser accedidos a través de una interfaz web.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.6: Requisito de Usuario - Requisito de Capacidad 05

RUC-06	
Nombre	Aplicación Android.
Descripción	Los datos del sistema podrán ser accedidos a través de una aplicación de la plataforma Android.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.7: Requisito de Usuario - Requisito de Capacidad 06

RUC-07	
Nombre	Acceso no centralizado.
Descripción	El sistema dará la posibilidad de acceder a los datos de una manera no centralizada.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.8: Requisito de Usuario - Requisito de Capacidad 07

RUC-08	
Nombre	Comunicación del sistema.
Descripción	Los dispositivos del sistema podrán comunicarse entre sí para obtener datos.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.9: Requisito de Usuario - Requisito de Capacidad 08

3.1.2. Requisitos de Restricción

RUR-01	
Nombre	Estándar de comunicación inalámbrica.
Descripción	El sistema se comunicará a través del estándar de comunicación inalámbrica wifi.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	RUC-08.

Tabla 3.10: Requisito de Usuario - Requisito de Restricción 01

RUR-02	
Nombre	Servidor del sistema.
Descripción	El servidor del sistema será compatible con algún sistema operativo instalable en una Raspberry Pi y estará desarrollado en la plataforma Node.js.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.11: Requisito de Usuario - Requisito de Restricción 02

RUR-03	
Nombre	Representación datos históricos.
Descripción	Los datos históricos recogidos por los sensores serán presentados en forma de gráfica.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	RUC-03, RUC-04, RUC-05, RUC-06.

Tabla 3.12: Requisito de Usuario - Requisito de Restricción 03

RUR-04	
Nombre	Nombre de los nodos.
Descripción	Los nodos del sistema que se asocien con los sensores tendrán como nombre en su punto de acceso: el prefijo “mgiot” seguido del nombre del sensor que representan.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	RUC-07.

Tabla 3.13: Requisito de Usuario - Requisito de Restricción 04

3.2. Casos de uso

En este apartado se van a describir las distintas posibles interacciones que presenta el sistema con el usuario. Este flujo queda definido por los requisitos anteriormente impuestos y serán representados en forma de diagramas de caso de uso. Esto permitirá una fácil interpretación sea cual sea el *background* del cliente. Además se incluye una descripción textual de cada uno de los casos de uso.

Debido a que el acceso al sistema es posible mediante tres formas distintas, se van a especificar los casos de uso para cada tipo de acceso. Los diferentes tipos de acceso son: a través de la interfaz web proporcionada por la Raspberry Pi, mediante la aplicación Android desarrollada para móvil o haciendo una conexión manual directamente al punto de acceso de cada uno de los nodos del sistema con un dispositivo móvil. Cada una de estas formas de acceso o escenarios se describen sus casos de uso.

3.2.1. Diagrama general de casos de uso

Las imágenes que se muestran a continuación representan las distintas posibles interacciones que tiene el usuario con el sistema en cada uno de los tres escenarios de uso posibles.

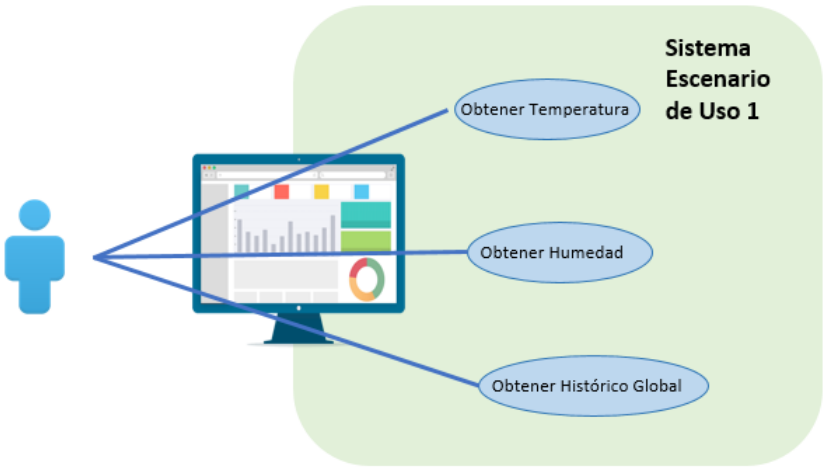

Figura 3.1: Diagrama general Casos de Uso Escenario 1

En el primer escenario uso, el usuario accede a la información del sistema a través de la interfaz web ofrecida por la Raspberry Pi. Figura 3.1

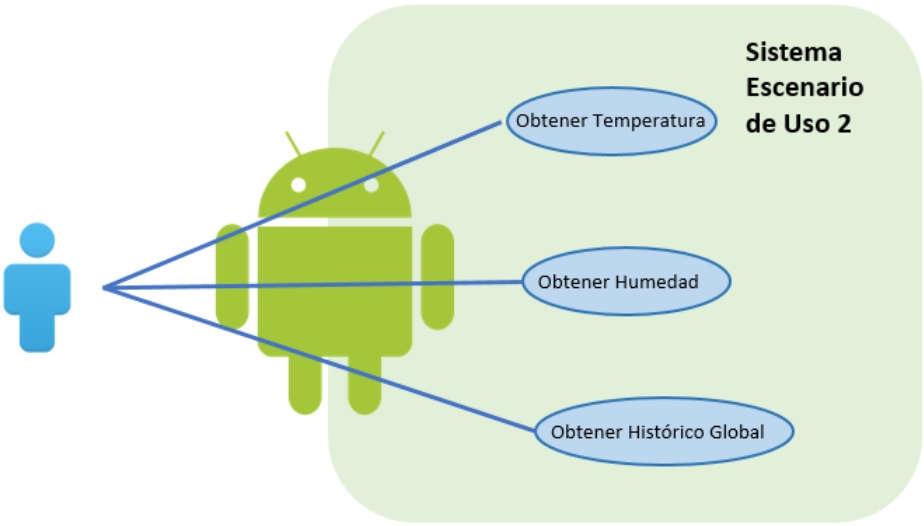

Figura 3.2: Diagrama general Casos de Uso Escenario 2

El segundo escenario de uso representa el acceso al sistema mediante la aplicación Android desarrollada para dispositivos móviles. Figura 3.2.

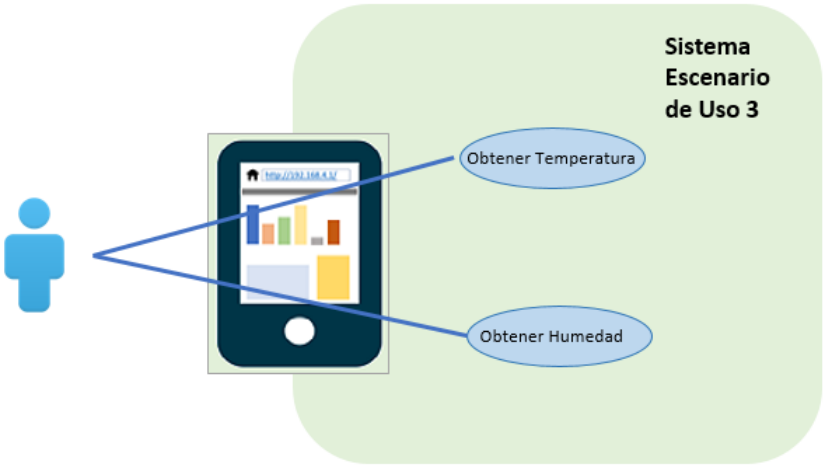

Figura 3.3: Diagrama general Casos de Uso Escenario 3

Finalmente, en el tercer escenario de uso el acceso al sistema lo hace el usuario de forma manual conectándose a cada uno de los nodos que componen el sistema de forma individual. Este acceso se lleva a cabo a través de cualquier dispositivo que tenga la capacidad de conexión wifi y disponga de navegador web. Figura 3.3

3.2.2. Descripción textual de casos de uso

A continuación, se listan los distintos casos de uso representados en el diagrama anterior. Para ello, se va a hacer uso de una tabla que sigue la siguiente estructura:

CU-EUX-XX	
Nombre	
Actores	
Descripción	
Precondiciones	
Flujo principal	
Flujo alternativo	
Postcondiciones	

Tabla 3.14: Caso de Uso - Ejemplo

Los campos contenidos en la tabla 3.14, anteriormente mostrada, se definen de la siguiente manera:

- **Identificador:** determina inequívocamente cada uno de los casos de uso representados. La estructura del identificador se compone de tres partes y tiene la siguiente forma: CU-EX-XX La primera parte del identificador permite reconocer que se trata de un caso de uso, la segunda representa el escenario al que pertenece y la tercera representa el número del caso de uso.

- **Nombre:** nombre asignado al caso de uso en particular. Explicará genéricamente el objetivo de dicho caso de uso.
- **Actores:** representa los elementos que participan.
- **Descripción:** breve desarrollo del escenario.
- **Precondiciones:** requisitos que se tienen que dar previamente a la ejecución del caso de uso.
- **Flujo principal:** se narra el caso de uso en forma de una secuencia de acciones, especificando qué elemento lleva a cabo cada una.
- **Flujo alternativo:** se detallan otros posibles flujos dentro del mismo caso de uso.
- **Postcondiciones:** se representan las variaciones que ha causado en el sistema el caso de uso.

3.2.2.1. Casos de Uso del Escenario de Uso 1

Los casos de uso representados a continuación son aquellos que se corresponden al primer escenario planteado, en el cual el usuario accede a la información a través de la interfaz web proporcionada por el servidor contenido en la Raspberry Pi.

CU-EU1-01	
Nombre	Obtener valores de temperatura.
Actores	Usuario.
Descripción	El usuario adquiere el valor de la temperatura actual en el entorno y un registro histórico de los últimos siete días.
Precondiciones	Estar encendido el nodo que recoge el valor de la temperatura, estar iniciado el servidor de la Raspberry Pi y levantada la interfaz web que el servidor proporciona.
Flujo principal	1. El usuario pulsa sobre la sección “Temperatura” de la interfaz web. 2. El servidor actualiza el valor actual de la temperatura con el último valor recuperado del nodo que recoge los valores de temperatura y envía los valores de temperatura recogidos en los últimos siete días.
Flujo alternativo	1. El usuario navega a la dirección url asociada a “Temperatura”. 2. El servidor actualiza el valor actual de la temperatura con el último valor recuperado del nodo que recoge los valores de temperatura del entorno y envía los valores de temperatura recogidos en los últimos siete días.
Postcondiciones	La interfaz web cambia su estado para mostrar el valor actual de la temperatura y presenta la variación de valores de temperatura recogidos en los últimos siete días en forma de gráfica.

Tabla 3.15: Caso de Uso CU-E1-01

CU-EU1-02	
Nombre	Obtener valores de humedad.
Actores	Usuario.
Descripción	El usuario adquiere el valor de la humedad actual en el entorno y un registro histórico de los últimos siete días.
Precondiciones	Estar encendido el nodo que recoge el valor de la humedad, estar iniciado el servidor de la Raspberry Pi y levantada la interfaz web que el servidor proporciona.
Flujo principal	1. El usuario pulsa sobre la sección “Humedad” de la interfaz web. 2. El servidor actualiza el valor actual de la humedad con el último valor recuperado del nodo que recoge los valores de humedad del entorno y envía los valores de humedad recogidos en los últimos siete días.
Flujo alternativo	1. El usuario navega a la dirección url asociada a “Humedad”. 2. El servidor actualiza el valor actual de la humedad con el último valor recuperado del nodo que recoge los valores de humedad del entorno y envía los valores de humedad recogidos en los últimos siete días.
Postcondiciones	La interfaz web cambia su estado para mostrar el valor actual de la humedad y presenta la variación de valores de temperatura recogidos en los últimos siete días en forma de gráfica.

Tabla 3.16: Caso de Uso CU-E1-02

CU-EU1-03	
Nombre	Obtener histórico global de datos.
Actores	Usuario.
Descripción	El usuario adquiere un registro histórico de cada uno de los datos que recoge el sistema.
Precondiciones	Estar iniciado el servidor de la Raspberry Pi y levantada la interfaz web que el servidor proporciona.
Flujo principal	1. El usuario accede a la página principal de la interfaz web. 2. El servidor vuelca los datos recopilados de los sensores en la página web.
Flujo alternativo	Ninguno.
Postcondiciones	La interfaz web muestra la variación de los valores recogidos en los últimos siete días de los diferentes sensores en forma de gráficos.

Tabla 3.17: Caso de Uso CU-E1-03

3.2.2.2. Casos de Uso del Escenario de Uso 2

Los casos de uso en el segundo escenario representan el acceso del usuario al sistema haciendo uso de la aplicación Android desarrollada para dispositivos móviles. Esta aplicación se comunica directamente con el servidor de la Raspberry Pi para obtener los valores de los distintos sensores del sistema.

CU-EU2-01	
Nombre	Obtener registro histórico de temperatura.
Actores	Usuario.
Descripción	El usuario adquiere el registro histórico de valores de temperatura en los últimos siete días.
Precondiciones	Estar iniciado el servidor de la Raspberry Pi y estar iniciada la aplicación Android con el wifi conectado a la misma red que el servidor.
Flujo principal	1. El usuario pulsa sobre la sección “Temperatura” de la interfaz de la aplicación. 2. La aplicación Android devuelve los datos históricos de temperatura recogidos por el sensor.
Flujo alternativo	Ninguno.
Postcondiciones	La interfaz de la aplicación muestra la variación de los valores de temperatura recogidos por el sensor en los últimos siete días en forma de gráfica.

Tabla 3.18: Caso de Uso CU-E2-01

CU-EU2-02	
Nombre	Obtener registro histórico de humedad.
Actores	Usuario.
Descripción	El usuario adquiere el registro histórico de valores de humedad en los últimos siete días.
Precondiciones	Estar iniciado el servidor de la Raspberry Pi y estar iniciada la aplicación Android con el wifi conectado a la misma red que el servidor.
Flujo principal	1. El usuario pulsa sobre la sección “Humedad” de la interfaz de la aplicación. 2. La aplicación Android devuelve los datos históricos de humedad recogidos por el sensor.
Flujo alternativo	Ninguno.
Postcondiciones	La interfaz de la aplicación muestra la variación de los valores de humedad recogidos por el sensor en los últimos siete días en forma de gráfica.

Tabla 3.19: Caso de Uso CU-E2-02

CU-EU2-03	
Nombre	Obtener histórico global de datos.
Actores	Usuario.
Descripción	El usuario adquiere un registro histórico de cada uno de los datos que recoge el sistema.
Precondiciones	Estar iniciado el servidor de la Raspberry Pi y estar iniciada la aplicación Android con el wifi conectado a la misma red que el servidor.
Flujo principal	1. El usuario pulsa sobre la sección “Servidor” de la interfaz de la aplicación. 2. La aplicación Android vuelca todos los datos obtenidos de los diferentes sensores que han sido enviados por el servidor.
Flujo alternativo	Ninguno.
Postcondiciones	La interfaz de la aplicación muestra la variación de los valores recogidos en los últimos siete días de los diferentes sensores en forma de gráficos.

Tabla 3.20: Caso de Uso CU-E2-03

3.2.2.3. Casos de Uso del Escenario de Uso 3

En el último escenario, el usuario realiza un acceso manual a cada uno de los nodos que contiene el sistema. Para ello, conecta un dispositivo móvil a la red wifi que ofrece cada uno de los puntos de acceso e introduce la dirección web del mismo para recoger el valor actual del sensor.

CU-EU3-01	
Nombre	Obtener temperatura.
Actores	Usuario.
Descripción	El usuario adquiere el valor de la temperatura actual en el entorno.
Precondiciones	Estar encendido el nodo que recoge el valor de la temperatura, disponer de un dispositivo móvil con conexión wifi y navegador y estar conectado el wifi del dispositivo móvil al punto de acceso ofrecido por el nodo.
Flujo principal	1. El usuario abre el navegador web del dispositivo y accede a la dirección web: http://192.168.4.1. 2. El nodo devuelve en formato de texto el último valor de temperatura recogido por el sensor.
Flujo alternativo	Ninguno.
Postcondiciones	En el navegador aparece una sencilla página que muestra el último valor recogido por el sensor y un botón que permite actualizarlo.

Tabla 3.21: Caso de Uso CU-E3-01

CU-EU3-02	
Nombre	Obtener humedad.
Actores	Usuario.
Descripción	El usuario adquiere el valor de la humedad actual en el entorno.
Precondiciones	Estar encendido el nodo que recoge el valor de la humedad, disponer de un dispositivo móvil con conexión wifi y navegador y estar conectado el wifi del dispositivo móvil al punto de acceso ofrecido por el nodo.
Flujo principal	1. El usuario abre el navegador web del dispositivo y accede a la dirección web: http://192.168.4.1. 2. El nodo devuelve en formato de texto el último valor de humedad recogido por el sensor.
Flujo alternativo	Ninguno.
Postcondiciones	En el navegador aparece una sencilla página que muestra el último valor recogido por el sensor y un botón que permite actualizarlo.

Tabla 3.22: Caso de Uso CU-E3-02

3.3. Requisitos Software

Una vez hecho el análisis de los requisitos de usuario y los caso de uso, se detallan los requisitos software derivados. Éstos tienen un carácter más técnico y describen la funcionalidad del sistema. Se dividen en dos tipos:

- **Requisitos funcionales:** describen las capacidades del sistema.
- **Requisitos no funcionales:** representan las propiedades o restricciones del sistema.

Los requisitos software se van a representar en una tabla con la siguiente estructura:

RS-XX	
Nombre	
Descripción	
Fuente	
Necesidad	
Prioridad	
Pre-requisito	

Tabla 3.23: Requisito de Software - Ejemplo

Los campos contenidos en la anterior tabla, figura 3.23, se detallan a continuación:

- **Identificador:** este valor identifica inequívocamente cada uno de los requisitos desarrollados. La estructura general del identificador sigue la siguiente forma: RS-XX. Está compuesto por dos partes: la primera indica que se trata de un requisito de software (RS) y la segunda determina el número que identifica al requisito. Al prefijo se le añade una "F" si es un requisito funcional y "NF" si se trata de un requisito no funcional.
- **Nombre:** nombre asociado con el requisito. Sirve para dar una idea general del objetivo del mismo.
- **Descripción:** desarrollo detallado del requisito de software.
- **Fuente:** indica qué actor ha propuesto el requisito en cuestión. Los actores pueden ser: cliente o jefe de proyecto.
- **Necesidad:** representa el grado en el cual se necesita que sea llevado a cabo el requisito. Los posibles valores son: opcional, deseable y esencial.
- **Prioridad:** indica el nivel de urgencia de que esté presente el requisito en el sistema. Los valores posibles son: baja, media y alta.
- **Pre-requisito:** en esta sección se indican otros requisitos que son necesarios para llevar a cabo el requisito en cuestión.

3.3.1. Requisitos Funcionales

RSF-01	
Nombre	Valor de la temperatura en tiempo real.
Descripción	El sistema podrá devolver en tiempo real el valor de la temperatura ambiente del entorno.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.24: Requisito Software - Requisito Funcional 01

RSF-02	
Nombre	Valor de la humedad en tiempo real.
Descripción	El sistema podrá dar en tiempo real el valor de la humedad del entorno.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.25: Requisito Software - Requisito Funcional 02

RSF-03	
Nombre	Valor histórico de la temperatura.
Descripción	El sistema podrá obtener un registro histórico de la temperatura ambiente en el entorno en los siete últimos días.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.26: Requisito Software - Requisito Funcional 03

RSF-04	
Nombre	Valor histórico de la humedad.
Descripción	El sistema podrá obtener un registro histórico de la humedad en el entorno en los siete últimos días.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.27: Requisito Software - Requisito Funcional 04

RSF-05	
Nombre	Interfaz web.
Descripción	Los datos del sistema podrán ser accedidos a través de una interfaz web.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.28: Requisito Software - Requisito Funcional 05

RSF-06	
Nombre	Aplicación Android.
Descripción	Los datos del sistema tendrán la posibilidad ser accedidos a través de una aplicación de la plataforma Android.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.29: Requisito Software - Requisito Funcional 06

RSF-07	
Nombre	Nodos del sistema.
Descripción	Los nodos del sistema que estén asociados con un sensor Tendrán que ofrecer un punto de acceso.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.30: Requisito Software - Requisito Funcional 07

RSF-08	
Nombre	Comunicación del sistema.
Descripción	Los dispositivos del sistema serán capaces de comunicarse entre sí para obtener datos.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.31: Requisito Software - Requisito Funcional 08

3.3.2. Requisitos No Funcionales

3.3.2.1. Operatividad

RSNF-01	
Nombre	Sistema operativo del servidor.
Descripción	El servidor alojado en la Raspberry Pi tendrá como sistema operativo Ubuntu Mate.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.32: Requisito Software - Requisito No Funcional 01

RSNF-02	
Nombre	Lenguaje del servidor.
Descripción	El servidor de la Raspberry Pi estará desarrollado en Node.js.
Fuente	Cliente.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.33: Requisito Software - Requisito No Funcional 02

RSNF-03	
Nombre	Base de datos del sistema.
Descripción	La base de datos del sistema será implementada en MongoDB.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.34: Requisito Software - Requisito No Funcional 03

RSNF-04	
Nombre	Lenguaje de la aplicación Android.
Descripción	La aplicación Android será desarrollada en el IDE Android Studio utilizando Java como lenguaje principal.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.35: Requisito Software - Requisito No Funcional 04

RSNF-05	
Nombre	Lenguaje de NodeMCU.
Descripción	Los microcontroladores del sistema se desarrollarán en el IDE de Arduino utilizando como lenguaje principal Arduino .
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.36: Requisito Software - Requisito No Funcional 05

3.3.2.2. Interoperabilidad

RSNF-06	
Nombre	Comunicación entre dispositivos.
Descripción	La comunicación entre los diferentes dispositivos del sistema se llevará a cabo usando el estándar de conexión inalámbrica wifi.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.37: Requisito Software - Requisito No Funcional 06

RSNF-07	
Nombre	Paso de datos entre dispositivos.
Descripción	La información enviada entre los distintos dispositivos que no sean los nodos del sistema tendrá formato JSON.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.38: Requisito Software - Requisito No Funcional 07

RSNF-08	
Nombre	Mensaje UDP del servidor.
Descripción	El servidor mandará un datagrama UDP en broadcast por la red local a través del puerto 11111 para que la aplicación Android reconozca su dirección IP.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.39: Requisito Software - Requisito No Funcional 08

3.3.2.3. Accesibilidad

RSNF-09	
Nombre	Acceso a la interfaz web de los nodos del sistema.
Descripción	La página web de los nodos del sistema será accedida a través de la dirección *http:192.168.4.1*.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.40: Requisito Software - Requisito No Funcional 09

RSNF-10	
Nombre	Interfaz web en el servidor.
Descripción	La interfaz web en el servidor de la Raspberry Pi será accedida a través de la dirección *localhost* y puerto 3000.
Fuente	Jefe de Proyecto.
Necesidad	Esencial.
Prioridad	Alta.
Pre-requisito	Ninguno.

Tabla 3.41: Requisito Software - Requisito No Funcional 10

3.3.2.4. Interfaz

RSNF-11	
Nombre	Presentación de datos históricos.
Descripción	La presentación de los datos históricos recopilados por los sensores se hará mediante el uso de gráficos.
Fuente	Jefe de Proyecto.
Necesidad	Deseable.
Prioridad	Media.
Pre-requisito	Ninguno.

Tabla 3.42: Requisito Software - Requisito No Funcional 11

3.3.2.5. Usabilidad

RSNF-12	
Nombre	Facilidad de uso.
Descripción	Obtener la información del sistema en cualquiera de las aplicaciones será un proceso que cuente con menos de 5 pasos.
Fuente	Cliente.
Necesidad	Opcional.
Prioridad	Media.
Pre-requisito	Ninguno.

Tabla 3.43: Requisito Software - Requisito No Funcional 12

3.4. Matrices de Trazabilidad

Una vez planteados los requisitos y casos de uso del sistema, se van a relacionar en matrices de trazabilidad para comprobar su coherencia. Se van a relacionar los requisitos de usuario con los casos de uso, los requisitos de usuario con los requisitos de software funcional y los requisitos de software funcional con los casos de uso.

3.4.1. Matriz de Trazabilidad Requisitos de Usuario frente Casos de Uso

Casos de Uso	CU-E1-01	CU-E1-02	CU-E1-03	CU-E2-01	CU-E2-02	CU-E2-03	CU-E3-01	CU-E3-02
Requisitos de Usuario de Capacidad								
RUC-01	X			X			X	
RUC-02		X			X			X
RUC-03			X			X		
RUC-04			X			X		
RUC-05	X	X	X				X	X
RUC-06				X	X	X		
RUC-07							X	X
RUC-08	X	X	X	X	X	X		
Requisitos de Usuario de Restricción								
RUR-01	X	X	X	X	X	X	X	X
RUR-02	X	X	X					
RUR-03	X	X	X	X	X			
RUR-04							X	X

Tabla 3.44: Matriz de Trazabilidad Requisitos de Usuario frente Casos de Uso

3.4.2. Matriz de Trazabilidad Requisitos de Usuario frente Requisitos Software Funcionales

Requisitos Software Funcionales	RSF-01	RSF-02	RSF-03	RSF-04	RSF-05	RSF-06	RSF-07	RSF-08
Requisitos de Usuario de Capacidad								
RUC-01	X							
RUC-02		X						
RUC-03			X					
RUC-04				X				
RUC-05					X			
RUC-06						X		
RUC-07							X	
RUC-08								X
Requisitos de Usuario de Restricción								
RUR-01								X
RUR-02	No Aplica							
RUR-03			X	X	X	X		
RUR-04							X	

Tabla 3.45: Matriz de Trazabilidad Requisitos de Usuario frente Requisitos Software Funcionales

3.4.3. Matriz de Trazabilidad Requisitos Software Funcionales frente Casos de Uso

Casos de Uso	CU-E1-01	CU-E1-02	CU-E1-03	CU-E2-01	CU-E2-02	CU-E2-03	CU-E3-01	CU-E3-02
Requisitos Software Funcionales								
RSF-01	X						X	
RSF-02		X						X
RSF-03	X			X				
RSF-04		X			X			
RSF-05	X	X	X				X	X
RSF-06				X	X	X		
RSF-07							X	X
RSF-08	X	X	X	X	X	X		

Tabla 3.46: Matriz de Trazabilidad Requisitos Software Funcionales frente Casos de Uso

Capítulo 4

Diseño del Entorno

Una vez hecho el análisis del sistema y revisada la coherencia de los requisitos entre sí se va a dar paso a explicar el diseño final del sistema. Primero, se explicará el contexto en el que va a existir este proyecto, para así entender los diferentes dispositivos que van a estar comunicándose entre sí. Posteriormente, se incluirá un apartado en el que se describe la arquitectura de componentes físicos que va a tener el sistema. Para finalizar este capítulo, se describirá en detalle las funcionalidad y estructura software que posee cada componente dentro del sistema.

4.1. Contexto del Entorno

El proyecto planteado es un sistema enfocado en *Internet of Things* (IoT), por lo que su ubicación ideal será una zona que ofrezca una buena conectividad wifi y no haya interferencias en las bandas de 2.4Ghz o inhibidores que interfieran en las comunicaciones inalámbricas. La temática del trabajo es una estación meteorológica que recoja datos del ambiente en el que se encuentra el sistema, por lo que el entorno ideal sería una zona donde los datos recogidos tengan un valor que o bien varíe de manera notable en el tiempo o que no sea menospreciable para que de esta manera el sistema proporcione datos de interés y utilidad.

Dentro de este mismo sistema se van a encontrar tres escenarios diferentes de uso, para así dotar a los usuarios de mayores posibilidades a la hora de acceder a la información.

- **Primer escenario de uso:** En esta aproximación el usuario accede a la información a través de la interfaz web proporcionada por una Raspberry Pi.
- **Segundo escenario de uso:** En este segundo enfoque, el usuario obtiene los datos recogidos por el sistema haciendo uso de la aplicación Android desarrollada para este uso.

- **Tercer escenario de uso:** En este último caso, el usuario puede conectarse su dispositivo móvil directamente en la red ofrecida por cada uno de los nodos del sistema para obtener el valor en tiempo real.

4.2. Arquitectura del Sistema

La disposición que van a tener los componentes del sistema se muestra en la figura 4.1. En ella vienen representados los tres escenarios posibles que tiene un usuario a la hora de acceder.

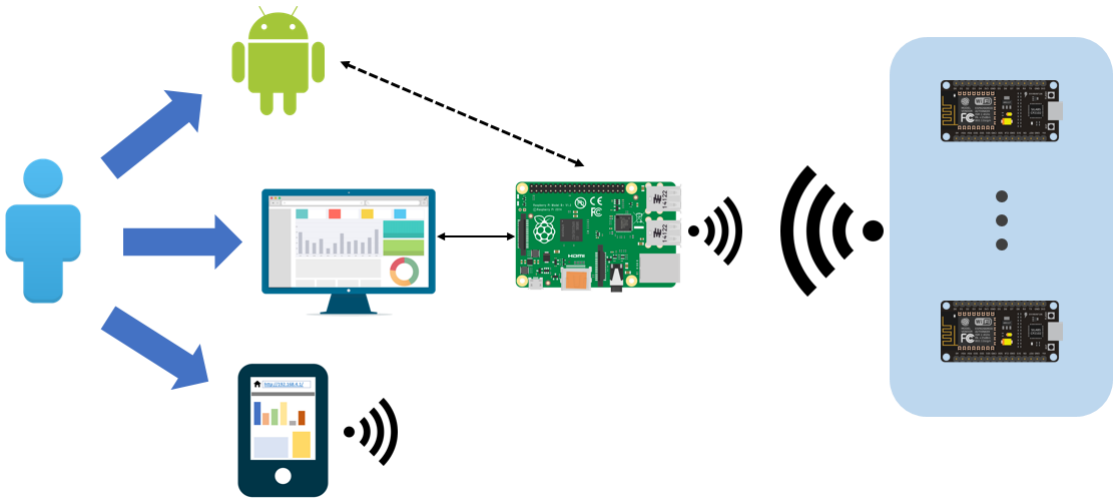

Figura 4.1: Arquitectura general del entorno

En la imagen mostrada se representa un usuario con tres formas posibles de acceso a la información contenida en los nodos.

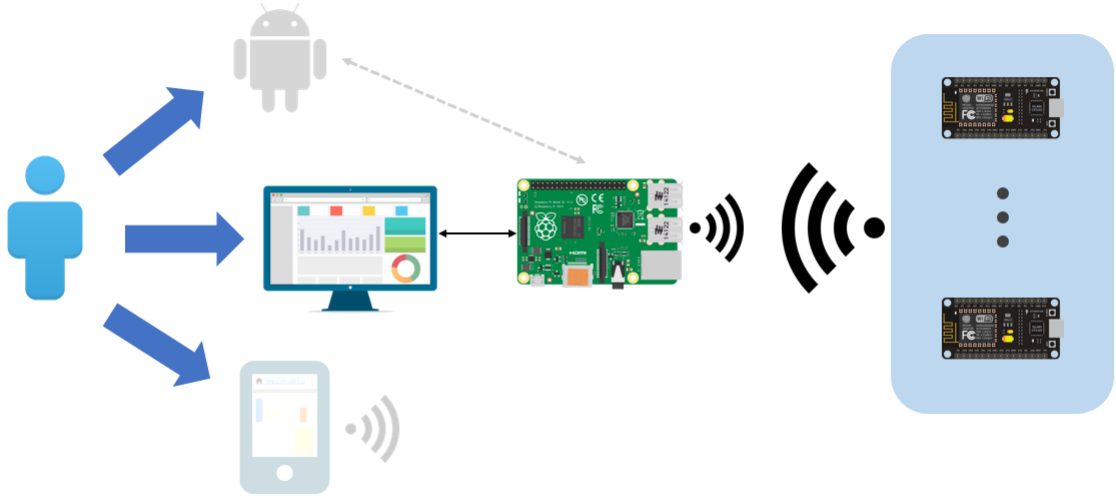

Figura 4.2: Arquitectura del primer escenario de uso

El primer escenario de uso, que se corresponde con la figura 4.2, presenta una interfaz web ofrecida por el servidor de la Raspberry Pi que contiene una serie de gráficos que representan los valores de los nodos.

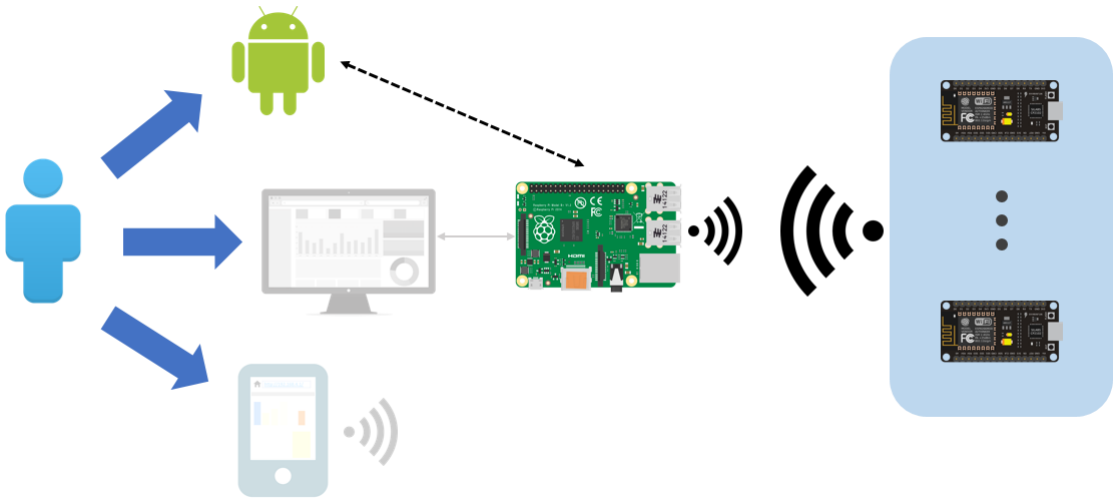

Figura 4.3: Arquitectura del segundo escenario de uso

La segunda opción, mostrada en la imagen 4.3, ofrece a los usuarios la posibilidad de utilizar la aplicación Android desarrollada para acceder al sistema. Esta aplicación se comunica directamente con el servidor de la Raspberry Pi para obtener los datos recopilados. La información se muestra en forma de gráficos de la misma forma que lo hace la interfaz web.

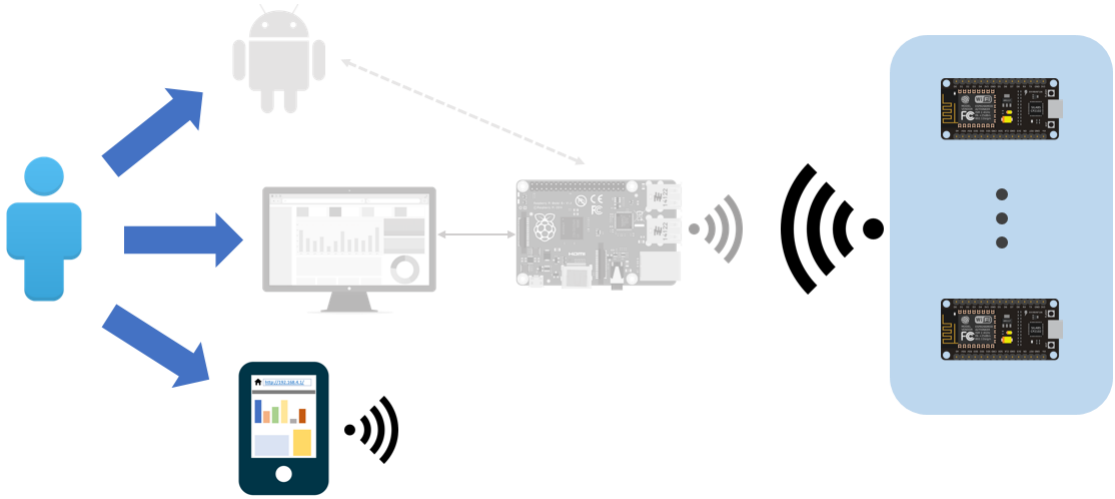

Figura 4.4: Arquitectura del tercer escenario de uso

La tercera y última opción de acceso al sistema es haciendo uso del navegador de un dispositivo móvil, como se representa en la figura 4.4. El dispositivo móvil se conecta a la red ofrecida por cada uno de los nodos y navega hasta la dirección web del mismo (que por configuración será fijada en:

http://192.168.4.1). En esta página web se muestra el valor que está recogiendo en tiempo real el sensor asociado al nodo.

4.3. Componentes de la Aplicación

En este apartado se describe la función de cada componente del sistema, detallando para ello todos los componentes software que contiene. En primer lugar se va a explorar la Raspberry Pi en su faceta como cliente-servidor que da soporte a la interfaz web. Seguidamente, se describe la aplicación Android desarrollada que se comunica con el servidor de la Raspberry Pi. Por último, se detalla el funcionamiento de los nodos del sistema formados por las NodeMCU ESP8266.

4.3.1. Raspberry Pi

La función principal de la Raspberry Pi es la de actuar como nodo central en los dos primeros escenarios de uso. Este ordenador se encarga de recopilar los datos ofrecidos por los nodos, almacenarlos de forma permanente y ofrecer una interfaz web para que los usuarios accedan a ella. Es por ello que este dispositivo ejercerá los roles de cliente y servidor de manera simultánea. Actuará como cliente cuando solicite datos a los nodos que recogen datos y de servidor cuando este proporcione páginas web para su visualización en un navegador o cuando resuelva peticiones a la aplicación de Android.

4.3.1.1. Servidor

La aplicación será desarrollada en Node.js sobre el sistema operativo Ubuntu Mate. La estructura de la aplicación se implementará a través del framework Express.js, el cual proporciona los ficheros y estructura de directorios para alojar una aplicación web.

Dicho framework proporciona una aplicación web básica totalmente funcional como punto de partida. Se puede acceder a su contenido abriendo un navegadador en el que se introduzca la dirección *localhost* a través del puerto 3000, haciendo la url de la siguiente forma: *http://localhost:3000*. Estos valores que vienen por defecto indican que se está accediendo a la dirección IP local de la máquina (conocida por su dirección 127.0.0.1 o por su nombre *localhost*) y que se está usando el puerto 3000 para ello.

4.3.1.1.1 Base de datos

La base de datos del sistema será desarrollada en MongoDB es un tipo de base de datos no relacional. La abstracción de los datos en registros dentro de tablas ha sido convertida en documentos dentro de colecciones. La estructura para almacenar los documentos en las colecciones hace uso del formato *JavaScript Object Notation* conocido por el acrónimo JSON.

Su inclusión en la aplicación se realizará a través del middleware Monk, que ofrece una sencilla interfaz para interactuar con la base de datos. Utilizando el manejador que proporciona, el servidor va a crear una colección dedicada a cada uno de los nodos existentes en la red. Dentro de estas colecciones se almacenan los datos recogidos de los nodos. De la misma forma, los nodos del sistema quedarán registrados en una colección dedicada exclusivamente a almacenar sus nombres, para que el servidor sepa en todo momento qué nodos tiene que manejar.

Los datos que se almacenan en las colecciones de cada uno de los nodos tendrán la siguiente estructura:

- **ID:** valor autogenerado por el motor de la base de datos para la identificación única del documento en la colección.
- **Value:** este campo de tipo JSON recoge el valor númerico devuelto por el sensor como la temperatura o nivel de humedad. Sin embargo, podría ser almacenados valores cualitativos si estos son indicados por el sensor.
- **Time:** marca de tiempo que guarda la fecha, hora, minuto, segundos y milisegundos en la que se registró el valor.

Cada vez que se recibe un valor nuevo de un sensor es almacenado en su correspondiente colección con la estructura anteriormente detallada. A la hora de extraer los datos para enviarlos a la página web o a la aplicación Android, se lanza una consulta a la base de datos para que recoja los valores registrados en los últimos siete días, para formar las gráficas con el historial semanal.

4.3.1.1.2 Enrutamiento de direcciones

El diseño de una aplicación web es relativamente sencillo, pues se trata de ir vinculando las diferentes rutas del sitio web con los ficheros y datos que se devuelven a cada una de esas peticiones. La idea es configurar una página de tal forma que tenga una pantalla asociada a cada uno de los nodos del sistema y una pantalla general en la que los recoja todos al mismo tiempo.

En las pantallas individuales se mostrará el gráfico que contiene la varianza de los datos a lo largo de los últimos siete días, así como también se representa el valor que está leyendo el sensor en tiempo real. El valor en tiempo real que se muestre en estas pantallas se corresponderá con el último valor registrado por el sensor que ha sido almacenado en el servidor.

De igual forma, en la pantalla general se mostrarán las gráficas que representan la varianza de los datos en la última semana, de cada uno de los nodos del sistema, pero sin mostrar el valor en tiempo real de cada nodo.

4.3.1.1.3 Cambio de redes

El objetivo de este apartado es recoger los datos que ofrecen los sensores y permitir el acceso de las aplicaciones Android a la información contenida en el servidor. Para ello, la Raspberry Pi tiene que cambiar de red wifi, conectándose a los puntos de acceso ofrecidos por los nodos y a la red wifi local establecida. Por tanto, el servidor tiene que estar haciendo periódicamente una función de barrido de redes wifi, almacenando aquellas redes de los nodos que pertenezcan al sistema (para identificarlas se decidió fijar el prefijo “mgiot” seguido del tipo de datos que proporciona como temperatura, humedad) e ir conectándose una a una para obtener su valor. La red wifi local establecida tendrá un nombre determinado previamente. Este proceso deberá separarse en dos funciones distintas: una primera que escanee los puntos de acceso presentes en el entorno y una segunda función que ejecute su conexión a cada una de las redes encontradas.

La primera función sirve de escaneo de los puntos de acceso, la cual se realizará en intervalos. Se ha decidido hacerlo de esta manera para convertir la función de escaneo en una función de tipo asíncrona y así conservar el funcionamiento del servidor. Dentro de este método se escanean las redes del entorno y se almacenan aquellas que tengan el prefijo “mgiot”. Estas redes se registran en la base de datos si no estaban presentes previamente. De la misma forma, se almacena la red wifi local a la que se van a conectar los dispositivos Android, que para el caso de este proyecto será la red wifi de casa. Al finalizar el proceso se invoca a la segunda función, que conecta el servidor con las redes almacenadas.

La segunda función conectará el servidor a cada una de las redes registradas en el proceso de escaneo. Tras hacer la conexión a la red en cuestión con éxito se pueden dar dos alternativas: la conexión se ha dado con uno de los nodos del sistema o la conexión ha sido a la red wifi local.

Si la conexión se hace a uno de los nodos del sistema, una vez se haya comprobado que la conexión ha sido exitosa, se procede a lanzar una petición HTTP de tipo GET, para obtener el valor del sensor. Una vez devuelto el valor, se almacena en su colección correspondiente en la base de datos. Esta operación que lleva a cabo el servidor hará que tome el papel de cliente en este prcoeso, pues es el que dirige la petición al nodo.

Si la conexión se hace a la red wifi de los dispositivos Android el proceso cambia totalmente. En esta conexión el servidor tiene que lanzar un mensaje a toda la red indicando que está activo para recibir sus peticiones. Este mensaje se lanza a través de broadcast para lo cual hay que utilizar un datagrama de tipo UDP. Se envía a través de la dirección IP de broadcast de la máquina y del puerto establecido. Una vez se haya enviado el mensaje, se aplicará un retraso de unos segundos al proceso antes de cambiar de red nuevamente. Esto se hace para darle el tiempo suficiente a los móviles de interpretar el datagrama y lanzar su petición al servidor antes de que éste cambie de red y ya no sea posible la comunicación.

4.3.2. Aplicación Android

La aplicación móvil de Android, cuyo nombre es “Jarvis”, permitirá acceder a la información de los nodos a través de una aplicación nativa. Esto contribuirá a reducir el tiempo de carga para la visualización de los datos debido a que tan solo serán enviados los datos obtenidos de los sensores desde el servidor a la aplicación móvil. Además, el acceso a la información será más fácil una vez que la aplicación esté instalada ya que con tan sólo con dos clicks se podrán visualizar las gráficas de datos sin tener que introducir una url en un navegador. Los servicios que se proporcionen en la aplicación web serán los mismos que los que ofrece la interfaz web en el sevidor de la Raspberry Pi, con la excepción de que no se muestra el valor en tiempo real de los nodos.

4.3.2.1. Recogida de datos

Para obtener los datos del servidor de la Raspberry Pi se utilizará el mensaje UDP que envía por la red. La interceptación de estos mensajes permitirá obtener la dirección IP del servidor y así se podrá dirigir una petición HTTP directamente.

Por tanto, en el inicio de la aplicación se debe iniciar un servicio que sea capaz de captar este tipo de mensajes en la red wifi. La aplicación quedará a la espera de recoger alguno de estos mensajes. Cuando se intercepte uno de ellos, la aplicación accederá a la dirección IP del emisor del mensaje, que es el servidor, y enviará una petición HTTP de tipo GET para obtener los datos de los nodos.

Cuando la petición sea respondida por el servidor se almacenarán los datos en el dispositivo. Existen varias formas de guardar esta información de modo que persista a lo largo del manejo de la aplicación por el usuario. Entre las diferentes posibilidades de almacenar los datos se encuentra: la creación de un fichero o la creación de una base de datos. La base de datos no parece muy buena opción, debido a que los datos que se usan en la aplicación son los que se obtienen del servidor cada vez que se abre la aplicación, por lo que no sería necesario llevar a cabo este proceso que seguramente consuma más recursos y tiempo que la creación de un fichero. La creación de un fichero se puede hacer utilizando la funcionalidad que ofrece Android de SharedPreferences, que crea un fichero de tipo clave-valor cuyo acceso puede ser restringido a la aplicación.

4.3.2.2. Representación de datos

Una vez se hayan obtenido y almacenado los datos del servidor se presentará una pantalla general que representa de manera gráfica cada uno de los nodos del sistema, incluyendo el propio servidor. La función de esta pantalla es la de indicar al usuario cuántos nodos hay presentes en el sistema que son accesibles. Si se pulsa cada uno de estos nodos, se abre una pantalla que muestra en una gráfica la varianza de los valores que ha tenido dicho sensor a lo largo de los últimos siete días. En caso de que el servidor sea pulsado, se mostrarán todas las gráficas con la varianza de los valores en la última semana de los sensores del sistema.

Es decir, en la pantalla general se van a representar una serie de imágenes que actúan como botones que van a permitir al usuario navegar entre la información de los distintos nodos y del servidor. Cuando se pinche una de las imágenes que representan los distintos sensores del sistema, se pasa a una nueva pantalla en la que se pinta una gráfica. Esta gráfica contiene la varianza de los datos recogidos por el sensor a lo largo de los últimos siete días. En caso de que la imagen del servidor fuera

presionada, se mostrarían tantas gráficas como tuviera el sistema, representando en cada una de ellas la varianza de los datos recogidos en la última semana.

4.3.3. Nodos del sistema

Los nodos del sistema van a ser los elementos encargados de proporcionar la información. El intercambio de datos se tiene que realizar a través de internet utilizando conexiones wifi, pues así tendrá un mayor número de usuarios que puedan acceder a su información. Esto permitirá que los usuarios puedan conectarse directamente al dispositivo sin necesidad de que haya un router que actúe como intermediario.

Para permitir que este acceso sea directamente al dispositivo utilizando la interfaz wifi se debe diseñar un servidor web. Este servidor será capaz de atender peticiones HTTP de tipo GET en las que podrá encapsular los valores obtenidos. El formato de los datos que se devuelvan en estas peticiones puede ser de diversas formas, cabiendo la posibilidad de poder retornar el valor como texto plano o incluirlo en una página web. La primera opción será ideal para el paso de datos a otros dispositivo no finales, como pueda ser el servidor de la Raspberry Pi, que manipulará este valor antes de mostralo al usuario. Sin embargo, la opción de devolver una página web permite que los usuarios puedan conectarse directamente al microcontrolador y, en vez de obtener un simple número en pantalla, se pueda mejorar la funcionalidad y diseño de dicha opción.

Por tanto, este servidor web será capaz de atender dos tipos de peticiones: una para dispositivos intermediarios y otra para dispositivos finales. El primer tipo de peticiones tendrá como respuesta el valor del sensor en texto plano, mientras que el segundo tipo de peticiones retornará el valor dentro de una página web.

Capítulo 5

Implementación

Tras diseñar cada uno de los componentes del sistema se da paso a este capítulo de implementación, que tiene como objetivo explicar el desarrollo de cada una de estas aplicaciones.

5.1. Raspberry Pi

Este ordenador que actúa como ordenador central de dos de los escenarios de uso del sistema se divide en dos componentes: una parte servidora y otra cliente. La parte servidora va a atender las peticiones enviadas por la interfaz web y las de la aplicación Android. La parte cliente es la encargada de recopilar los datos de los nodos del sistema.

Como se ha comentado en el capítulo anterior, el desarrollo de estas piezas estará hecho en Node.js haciendo uso del template Express.js. La página web utiliza el sistema de plantillas Jade, que a su vez tendrá código Javascript embebido.

5.1.1. Aplicación Servidor

La funcionalidad del servidor de la Raspberry Pi se divide en una serie de archivos dentro de la estructura creada por Express.js. El núcleo de la aplicación se concentra en el fichero denominado *app.js*, cuya ubicación es en la raíz de la carpeta. Las funciones encargadas de responder a las peticiones entrantes se encuentran dentro de la carpeta *routes*, que contiene diversos archivos de Javascript. Finalmente, los archivos que se corresponden a las vistas de la interfaz web, cuyo formato es Jade, se encuentran localizados dentro de la carpeta *views*.

5.1.1.1. Base de Datos

Una vez ha sido creada con éxito la base de datos del sistema se procede a vincularla con la aplicación del servidor. Dentro de la aplicación general del servidor (*app.js*) se llevan a cabo diversas operaciones con la base de datos.

La primera de ellas es obtener el manejador de la base de datos en la zona de declaración de variables, al principio del fichero. Para llevar esto a cabo se utiliza la librería Monk, la cual devuelve un manejador para interactuar con la base de datos. Este manejador será incluido en el objeto *request* de cada petición HTTP para que los ficheros encargados de atender dichas peticiones puedan tener acceso a la base de datos.

El siguiente acceso a la base de datos lo encontramos en la función que realiza el cambio de redes del servidor. Dentro de este método se accede a la colección que almacena los nombres de los nodos del sistema y registra aquellos nodos nuevos en el sistema que no están presentes en dicha colección. Para ello, se realiza una consulta a la colección con el nombre del punto de acceso al que se está accediendo y si el nombre no se encuentra en la colección se hace un nuevo registro.

La siguiente operación se encuentra en la función de obtención de datos de los nodos del sistema. Después de hacer la conexión con éxito a uno de los nodos del sistema, se manda una petición HTTP de tipo GET con la finalidad de obtener el último valor recogido por el sensor. Una vez se obtiene el dato se procede a insertarlo en la base de datos. Lo primero que se hace es obtener el manejador de la colección, la cual comparte el nombre con el del nodo. Con este manejador se hace una inserción introduciendo el valor recogido y la marca temporal de ese momento, que incluye fecha y hora.

En los ficheros que atienden a las peticiones web se encuentran más operaciones con la base de datos, pero prácticamente es la misma operación para todos ellos. Se pueden dar dos tipos de peticiones diferentes: una que pida el registro histórico de un nodo en particular y otra que pida el registro histórico de todos los nodos del sistema.

La primera petición se resuelve obteniendo el manejador de la colección del nodo especificado y lanzando una consulta a esa colección. Esta consulta está diseñada para que recoja aquellos valores cuya marca temporal quede dentro de los últimos siete días, con fecha de partida la del momento actual de la consulta.

La petición que atiende el registro histórico de todos los nodos del sistema tiene esta misma consulta, con la particularidad de que se repite el proceso tantas veces como nodos registrados haya en la colección que almacena los nombres de los nodos del sistema.

Los registros almacenados en la base de datos tendrán una estructura como en la figura 5.1.

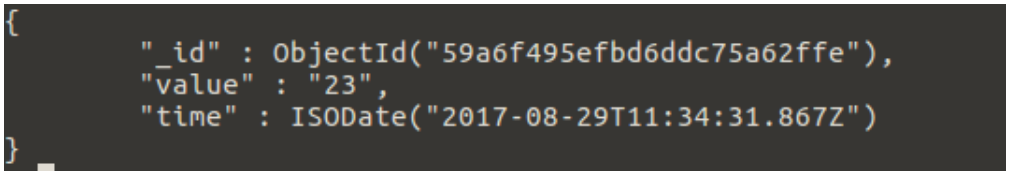

Figura 5.1: Ejemplo de registro en MongoDB

5.1.1.2. Enrutamento de direcciones

La estructura proporcionada por Expresss.js ya deja una ruta preparada para albergar la página general del sitio web (conocida comúnmente como *index* y cuya ruta asociada es: *http://localhost:3000*). Aprovechando esto, simplemente basta con modificar el método que devuelve el fichero asociado y el fichero en sí. Dentro del método se ha de incluir una sección de código en la que se recogen los datos de la base de datos. Para ello, se instancia un manejador de la base de datos y primero se lanza una consulta a la colección que recoge el nombre de los nodos del sistema. Una vez se tengan los nombres, se lanza una consulta a cada uno de ellos pidiendo los datos almacenados en la última semana. Cuando se tengan todos los datos recogidos en formato JSON, se renderiza la página *index.jade* con dicha información. Esta página está modificada para pintar una gráfica por cada nodo que reciba en la que se refleja la varianza de los datos en la última semana. Para pintar estas gráficas se utiliza la librería Chart.js.

El fichero que atiende las peticiones de la aplicación Android tiene la misma estructura que el *index.js*, con la excepción de que no se devuelve un fichero Jade, sino que se devuelven los datos en formato JSON. La aplicación se encarga de pintar las gráficas con su propia librería. La ruta de esta petición es: *http://localhost:3000/phone*. Como el móvil no puede acceder a la dirección *localhost* del servidor, se sustituye este parámetro por la dirección IP de la máquina.

El procedimiento para las páginas individuales es prácticamente el mismo, basta con añadir en el fichero *app.js* la ruta de la página web y asociarla con el método y fichero que se va a devolver. Las rutas de estas páginas individuales son: *http://localhost:3000/Temperatura* y *http://localhost:3000/Humedad*.

El proceso de recogida de datos de estos métodos va a ser ligeramente distinto, pues como no es necesario obtener el nombre de todos los nodos del sistema, simplemente basta con acceder a los datos contenidos en la colección del sensor en cuestión. En lo que sí difiere es en el envío de los valores en tiempo real de cada uno de los sensores, que la página principal no proporciona. Para llevarlo a cabo, se utiliza la librería Socket.io que abre un canal de comunicación bidireccional con las páginas específicas. El manejador que ofrece esta librería es asociado a una serie de eventos para que se ejecute una funcionalidad. En este caso, cada página específica va a activar un evento periódico en el servidor para recoger el último valor almacenado de su sensor, por lo que el servidor cada vez que obtenga un nuevo valor lo almacenará en una variable que quedará en espera de que se vuelva a lanzar este evento.

5.1.1.3. Cambio de redes

El proceso de cambio de redes que hace el servidor tiene dos objetivos: recoger información de los nodos y permitir el acceso a los datos a las aplicaciones Android. Para recoger la información de los nodos el servidor ha de conectarse al punto de acceso que éstos ofrecen, tras lo cual envía una petición HTTP de tipo GET a través de la cual obtiene el valor del sensor. El acceso de las aplicaciones Android al servidor se lleva a cabo mandando un datagrama UDP en broadcast por la red de wifi local establecida.

En primer lugar, se va a explicar cómo se realiza el cambio de redes wifi en el servidor. Este proceso se realiza en dos funciones: una primera que escanea y registra los nodos presentes en el sistema y otra función que se encarga de conectarse a los nodos registrados. Ambas funciones utilizan la librería WifiControl para manejar el wifi del ordenador.

La primera tarea que se ejecuta es el escaneo de redes wifi en la zona en la que se ubica el servidor. Tras este escaneo, se analiza cada uno de los nombres de las redes encontradas buscando aquellas que tengan como prefijo “mgiot”, lo cual indica que pertenecen al sistema. Las redes que cumplan esta condición son incluidas en un array, y de la misma forma se incluye en el array la red de wifi local establecida. En el desarrollo de este proyecto se ha asignado esta red local como la red wifi de casa. Adicionalmente, se registran aquellos nodos con prefijo “mgiot” que no estuvieran previamente en la base de datos, concretamente en la colección dedicada a almacenar los nombres de los nodos. Este método se ejecuta en intervalos de 10 segundos, con la intención de no bloquear la ejecución del servidor en una sección de código que se ejecuta continuamente.

La segunda función consiste en conectarse a cada una de las redes wifi almacenadas en el proceso anterior. Este método funciona de manera recursiva utilizando el array con los nodos almacenados para iterar, dándose el caso de parada cuando éste se encuentre vacío. Se utiliza la función de la librería WifiControl para ello. Si la conexión ha tenido éxito se pueden dar dos casos:

1. **Conexión a los nodos del sistema**. En este caso el servidor hace la función de cliente, pues su propósito es enviar una petición HTTP de tipo GET al nodo al que se ha conectado. Esta petición se dirige a la dirección IP en la que se ha establecido el servidor web del nodo, la cual es igual para todos ellos. Cuando el valor es devuelto por la petición se procede a almacenarlo en su colección correspondiente dentro de la base de datos.

2. **Conexión a la red wifi local**. Cuando el servidor se conecta a la red wifi local establecida se permite el acceso de las aplicaciones Android a los datos almacenados en él. Para ello, el servidor crea y envía un datagrama de tipo UDP en broadcast por la red, utilizando el puerto 11111 (seleccionado dentro del rango de puertos dinámicos o privados). Este mensaje es interceptado por las aplicaciones Android y obtienen la dirección IP del servidor, con la cual pueden enviar una petición HTTP al servidor para solicitar los datos de los nodos. El servidor se queda conectado a esta red de dos a tres segundos aproximadamente, para dar el tiempo suficiente a las aplicaciones Android para llevar el proceso a cabo.

Después de estos dos casos, se elimina el nodo al que se ha conectado el servidor del array y se vuelve a invocar a la misma función con el array reducido. En caso de que no haya más nodos pendientes de conectar se saldrá de la función.

5.2. Aplicación Android

La aplicación Jarvis va a tener una funcionalidad muy parecida a la de la interfaz web que proporciona el servidor de la Raspberry Pi, con la excepción de que no va a mostrar el valor en tiempo real de los nodos.

Ha sido desarrollada en el entorno de desarrollo propio de Android, que se llama Android Studio. El lenguaje de programación principal es Java, que se complementa con ficheros XML que representan el diseño de la aplicación. La aplicación ha sido desarrollada en la última versión de Android 7.0 Nougat. La estructura se puede dividir en dos partes: la primera es de recepción de datos y la segunda es la representación de los mismos.

5.2.1. Recogida de datos

Esta parte compone el inicio de la aplicación, en el cual se obtienen los datos desde el servidor de la Raspberry Pi. Se utiliza el mensaje que envía en broadcast el servidor para determinar que su estado es activo, y de esta forma se procede a enviar la petición de los datos.

La primera pantalla que se ve en la aplicación es una imagen que indica que se están cargando los datos. Al mismo tiempo se ha iniciado un servicio de escucha de mensajes UDP, que es el responsable de captar estos mensajes en broadcast del servidor. Este servicio levanta un proceso de escucha de mensajes UDP en la dirección IP de broadcast del dispositivo móvil y a través del puerto 11111, que ha sido elegido para este propósito.

La elección de utilizar datagramas de tipo UDP se debe a que los dispositivos móviles y el servidor de la Raspberry Pi no se conocen entre sí. Entonces, no encontraba ninguna otra forma general que pudiera dar servicio a una serie de móviles desconocidos para el servidor si no hacía uso de una difusión ancha o broadcast. Con esto se consigue que el servidor pueda indicar a una serie de dispositivos que está activo y puede ofrecer servicio a la par que indica de manera indirecta su dirección IP, que se encuentra contenida en el datagrama. Este datagrama es de tipo UDP en vez del tradicional TCP porque este último, a pesar de ofrecer confiabilidad en el envío, no dispone de la posibilidad de enviar un mensaje a varios destinatarios, pues está orientado a comunicación entre dos máquinas únicamente.

Cuando este servicio detecte el mensaje UDP enviado por el servidor, capta la dirección IP del mismo y formula una petición HTTP de recogida de datos. Una vez haya llegado la respuesta del servidor, se procede a almacenar estos datos en una variable del sistema, cerrar el servicio de escucha levantado e iniciar la pantalla general de muestra de datos.

Los datos son guardados en un SharedPreferences para ser accedidos por toda la aplicación durante su ejecución. La información recibida está en formato JSON.

5.2.2. Representación de datos

Después de almacenar los datos obtenidos por el servidor, la aplicación navega a la pantalla principal de la aplicación. En ella aparecen una serie de imágenes que representan cada uno de los nodos del sistema presentes en estos datos recogidos y aparece el icono del servidor. Estas imágenes son botones que permiten navegar entre la información de cada uno de los elementos.

El resultado de pulsar cada una de estas imágenes es abrir una nueva pantalla donde se presentan los datos asociados a ese elemento en forma de gráfica, para lo cual se utiliza la librería MPAndroidChart. Se recogen los datos asociados a ese sensor contenidos en el SharedPreferences y se hace el cálculo de la varianza a lo largo de los últimos siete días, para finalmente representarlo en una gráfica responsiva. En caso de que se pulse el servidor se muestran tantas gráficas como sensores haya presentes en los datos.

Se puede navegar a la pantalla principal utilizando la flecha que presenta la barra superior de la aplicación o bien utilizando la flecha de ir hacia atrás del mismo dispositivo móvil.

5.3. Nodos del sistema

Los nodos del sistema se componen de un microcontrolador NodeMCU ESP8266 y un sensor DHT11. El microcontrolador utilizado es de tipo open source e incluye el firmware necesario para portar un módulo de wifi, el cual se puede configurar como punto de acceso. Los sensores mencionados son capaces de interpretar los parámetros de temperatura y humedad del ambiente.

El desarrollo de esta aplicación se ha llevado a cabo utilizando el IDE de Arduino. El lenguaje que utiliza se llama también Arduino y es una variante del lenguaje C. Una de las funciones devuelve una página web que ha sido codificada en HTML. Se utiliza la librería Adafruit DHT para acceder a los valores de temperatura y humedad leídos por los sensores.

Dentro de la aplicación encontramos tres funciones. La primera que se va a definir es la función *setup*, que sirve para inicializar el servidor web y asociar las posibles rutas con las funciones que las van a atender. El servidor web tiene como nombre o SSID el prefijo “mgiot” seguido del sensor que aloja, y es accesible a través de la url: *http://192.168.4.1*.

La primera función que se encarga de responder peticiones de usuarios está encargada de responder las solicitudes del servidor de la Raspberry Pi. Esta función recoge el valor actual del sensor y lo devuelve en formato de texto.

La segunda función está diseñada para responder a las peticiones que hagan los usuarios a través de un navegador web desde un dispositivo móvil. La respuesta será un fichero HTML en el cual se muestre el último valor del sensor recogido y un botón que permita actualizar dicho dato. La página web se actualiza constantement cada 5 segundos.

Capítulo 6

Discusión

El objetivo de este capítulo es hacer visible y estudiar aspectos de la exploración del entorno de Internet de las Cosas que no está contemplado en los anteriores apartados.

A lo largo de este capítulo se analizarán y mencionarán las ventajas y desventajas de los diferentes enfoques propuestos para el acceso a la información de los sensores. Además se recapitularán algunas decisiones que se han tomando en la implementación del diseño a las que se incluirán opiniones personales desde el punto de vista del desarrollador. Esto no solo permitirá justificar desde el punto de vista técnico futuras implementaciones sino que expondrá dificultades con las que se podrán encontrar futuros desarrolladores.

6.1. Escenarios de Uso del Sistema

A lo largo de este documento se han mencionado los tres enfoques que se han desarrollado para la implementación de este sistema. El primer sistema está compuesto por un servidor alojado en la Raspberry Pi que ofrece una interfaz web para acceder a los datos. El segundo de ellos, es una aplicación Android que se comunica con el servidor anteriormente mencionado para mostrar los datos. El último de ellos, es un acceso directo al punto de acceso ofrecido por los nodos del sistema, que se puede realizar desde cualquier dispositivo móvil que disponga de navegador y wifi.

6.1.1. Raspberry Pi

Este subsistema está formado por un servidor alojado en la Raspberry Pi que ofrece una interfaz web para acceder a los datos. La obtención de los datos la lleva a cabo el servidor, el cual también ofrece la interfaz para mostrarlos. Por tanto, este sistema es de tipo centralizado basado en la Raspberry Pi como ordenador central.

Se quiso desarrollar este sistema porque es uno de los más tradicionales en cuanto a acceso a la información se refiere. La página que ofrece está alojada en el servidor de manera local pero se podría subir a un dominio de internet para que fuera accesible desde cualquier otra máquina. Esto proporcionaría un acceso global a los datos de un área localizada.

El principal inconveniente de este enfoque es la necesidad de ir cambiando de red wifi para obtener los datos de los nodos. Esta práctica retrasa mucho el tiempo de respuesta del servidor ya que, además de necesitar realizar un *handshake* para realizar la consulta, es necesario establecer conexión wifi en cada consulta a un nodo.

6.1.2. Aplicación Android

La aplicación Android también compone un subsistema centralizado con el servidor de la Raspberry Pi pero sustituye el acceso a la información a través de un navegador web por una aplicación móvil.

Es un sistema más moderno que se adapta mejor a la era actual que vivimos, en la que dominan los smartphones y las tablets. A pesar de seguir formando un sistema centralizado, permite la movilidad que estos dispositivos ofrece.

En este sistema, el inicio de la comunicación entre la aplicación y el servidor se hace mediante el envío y recepción de datagramas UDP. Este método puede no ser el más eficiente, debido a la sobrecarga que causa en la red, pero se adapta muy bien al problema que se quiere abarcar.

Inicialmente, la aplicación Android no conoce ni tiene por qué conocer la dirección IP de un servidor colocado en un entorno de este estilo. Por otra parte, el servidor tampoco conoce qué dispositivos van a establecer comunicación con él. Por tanto, es lógico pensar que el servidor puede mandar un mensaje no dirigido a todos los usuarios de una red para indicar su actividad, enviando al mismo tiempo su dirección IP.

Los puntos a favor de esta solución es que es eficiente y que la aplicación puede dar cobertura a varios entornos situados en diferentes lugares que tengan este mismo enfoque. El punto negativo que le veo a esta solución es el hecho de que la mayoría de routers públicos tienen desactivado este tipo de mensajes, por lo que la aplicación no podría ser portable a diversos lugares sin configurar previamente el router. El envío de datagramas se realiza cada varios segundos, por lo que no debería tener un gran impacto en la sobrecarga de la red.

Para esta solución se exploraron otras alternativas como el uso de Bluetooth o establecer un punto de acceso wifi en el dispositivo móvil. Ambas opciones fueron descartadas debido a que tenían poco o ningún soporte para llevar a cabo este tipo de operación.

6.1.3. Nodos del sistema

El último subsistema que se encuentra en el entorno sí es descentralizado y está compuesto por cada uno de los nodos que aportan la información al sistema.

Cada uno de los nodos ofrece un punto de acceso wifi al que se pueden conectar los usuarios con sus dispositivos móviles. Al conectarse a ellos, reciben una página web con el último valor leído por el nodo, que se actualiza automáticamente cada unos segundos.

El único inconveniente que encuentro a este sistema es el hecho de que el usuario tenga que saber la dirección web a la que se tiene que dirigir una vez que se haya conectado a la red wifi. Me parece bien que no haya un nodo central que agrupe estos datos, para eso está el servidor en los otros subsistemas.

6.1.4. Conclusión Final

Hasta el día de hoy, estas implementaciones son unas opciones válidas. Sin embargo, en un futuro cercano se espera que la proliferación de los dispositivos con tecnología *Wifi Direct* sea mayor y esto se traduzca en reducción de los precios de las placas que tiene un circuito integrado para este fin, y que así aumente su soporte para el desarrollo. Esto permitirá la exploración de enfoques que agrupen todas las ventajas de cada uno de los tres enfoques anteriores.

Capítulo 7

Gestión del Proyecto

El presente capítulo describe en detalle la planifcación que fue diseñada para la realización de este trabajo fin de grado así como el presupuesto final del mismo. A lo largo del primer apartado se explicará el ciclo de desarrollo seleccionado para el proyecto, la planificación inicial proyectada antes de empezar y la planificación final alcanzada tras varias replanificaciones. En la segunda parte del capítulo se especificarán los gastos derivados de la ejecución del proyecto.

7.1. Planificación

Dada la magnitud del tamaño de este proyecto software es necesario seleccionar un ciclo de desarrollo para este proyecto así como la realización de una planificación que permita identificar cada una de las tareas que serán necesarias y calcular una estimación de tiempo para cada una de ellas. Pese a que una buena planificación requiere una inversión inicial de tiempo, los beneficios que proporciona justifican con creces el tiempo invertido. Entre las ventajas que ofrece cabe mencionar:

- Obtención de un plan metódico y coherente que guía el trabajo diario.
- Posibilidad de conocer en cualquier momento el estado actual progreso que tiene el proyecto y el trabajo que queda por realizar.
- Toma de decisiones anticipadas y posibilidades de estudio de alternativas antes iniciar el proyecto.
- Cuantificación de los recursos necesarios (incluyéndose horas de trabajo) para conocer la viabilidad del proyecto.
- Reducción de la posibilidad de fracasar (*Failing to plan is planning to fail*).

7.1.1. Ciclo de Desarrollo

El proceso de desarrollo software que ha sido usado durante este proyecto ha sido el modelo en cascada en su variante de retroalimentación. Este modelo de proceso describe una serie de etapas que se han de llevar a cabo en riguroso orden, no permitiéndose iniciar una nueva fase del proceso hasta que todas las anteriores estén acabadas. La variante con retroalimentación permite retroceder en el proceso si se ha detectado algún fallo o incoherencia. Sin embargo, una vez resuelto el problema o incongruencia las etapas posteriores al problema han de ser modificadas, si es necesario, para continuar con el proceso. En la figura 7.1 se muestra el diagrama de ciclo de vida del desarrollo en cascada con retroalimentación.

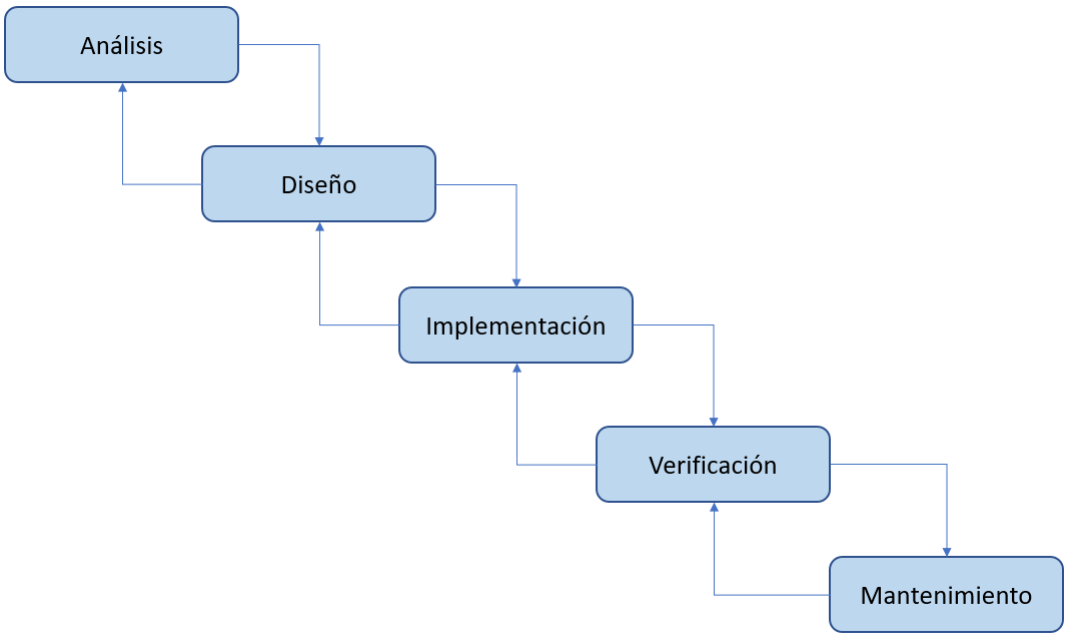

Figura 7.1: Ciclo de desarrollo en cascada retroalimentado

Las ventajas que ofrece usar este modelo son las siguientes:

- Estructura fácil de entender y seguir.
- Es uno de los modelos más comunes y utilizados en el desarrollo de software.
- La retroalimentación permite cierta flexibilidad a la hora de tratar con errores o incoherencias dentro del proyecto.

Por otra parte, este modelo de desarrollo tiene algunas desventajas como las listadas a continuación:

- Se tarda bastante tiempo en llegar hasta el producto final.
- No se puede llevar a cabo una etapa si la etapa previa no ha sido completada.
- Los fallos que se detectan en las últimas fases del proceso pueden provocar largos retrasos.
- Los usuarios finales no intervienen en las decisiones por lo que no se puede conocer su opinión hasta que el producto no ha sido terminado.

7.1.2. Planificación Previa

La planificación inicial del proyecto se llevó a cabo poco después de conocer la temática del trabajo. Esto se produjo en enero de 2017 tras un intercambio de correos y reuniones con el tutor. Sin embargo, debido a la coincidencia con el período de exámenes del primer cuatrimestre la planificación fue proyectada a partir de febrero y tenía como objetivo concluir el desarrollo y documentación antes de la convocatoria de julio. Las fases principales que fueron identificadas para la planificación fueron:

1. **Planificación:** el objetivo de este paso es identificar cada una de las etapas que son requeridas concluir este proyecto así como una estimación de tiempo necesaria para realizar cada una de ellas.
2. **Análisis del Problema Inicial:** este paso tiene como meta obtener un panorámica básica de las tecnologías actuales que son utilizadas en el área o campo sobre el que trabaja este proyecto. Al mismo tiempo, esto permite poder realizar un análisis de los problemas y limitaciones de la tecnología.
3. **Análisis del Proyecto:** esta etapa tiene como objetivo describir en detalle el sistema a implementar a través de requisitos de software. Para ello, esta fase incluye la realización de una serie de entrevistas con el profesor, que ejerce el rol de cliente, y su análisis para la extracción de los requisitos de usuario, la identificación de los casos de uso y la redacción y verificación de los requisitos software.
4. **Diseño:** el resultado final de esta fase son modelos, esquemas o diagramas que describan con un alto nivel de detalle los componente que formarán las piezas de software así como la arquitectura en las que estarán distribuidas. Las decisiones que se llevarán a cabo en el diseño tendrán

en cuenta el análisis del problema inicial y los requisitos obtenidos en la fase de análisis del proyecto

5. **Implementación:** esta etapa llevará a cabo la programación de cada uno de las piezas software que componen el sistema planteado en el diseño según las tecnologías seleccionadas.

6. **Verificación:** a lo largo de esta fase se realizará una serie de pruebas para comprobar el correcto funcionamiento de cada una de los componentes del sistema al mismo tiempo que se comprueba que se cumplen los objetivos fijados inicialmente en los requisitos.

7. **Documentación:** etapa que tiene como objetivo documentar y explicar en detalle todo las tareas que se han realizado para la culminación del proyecto.

Se muestra de maner agráfica en el siguiente diagrama de Gantt, la figura 7.1.

Tabla 7.1: Diagrama de Gantt Planificación Inicial

7.1.3. Planificación Final

La planificación es una estimación aproximada en un momento determinado de cuáles son las tareas a realizar y cuanto tiempo requerirán. Por ello, no siempre se logra alcanzar los objetivo debido a problemas que no fueron correctamente identificados en la planificación o por circunstancias externas a un proyecto.

Sin embargo, esto no quiere decir que una planificación no sea una correcta manera o herramienta de llevar un proyecto de grandes dimensiones. En caso de que sucedan estas situaciones, se hace necesario realizar una replanificación que tenga en cuenta los problemas encontrados. A pesar de que la planificación de este proyecto fue proyectada de manera realista, no se pudo realizar en los tiempos acordados por una serie de eventos externos al proyecto. Por todo ello, se hizo necesaria una serie de replanificaciones que fueran conscientes de las dificultadas descubiertas.

El proyecto inicialmente fue programado para su finalización antes de la convocatoria de julio, sin embargo, tuvo que ser pospuesta su finalización antes de la convocatoria de septiembre. En la figura 7.2 se muestra la planificación final:

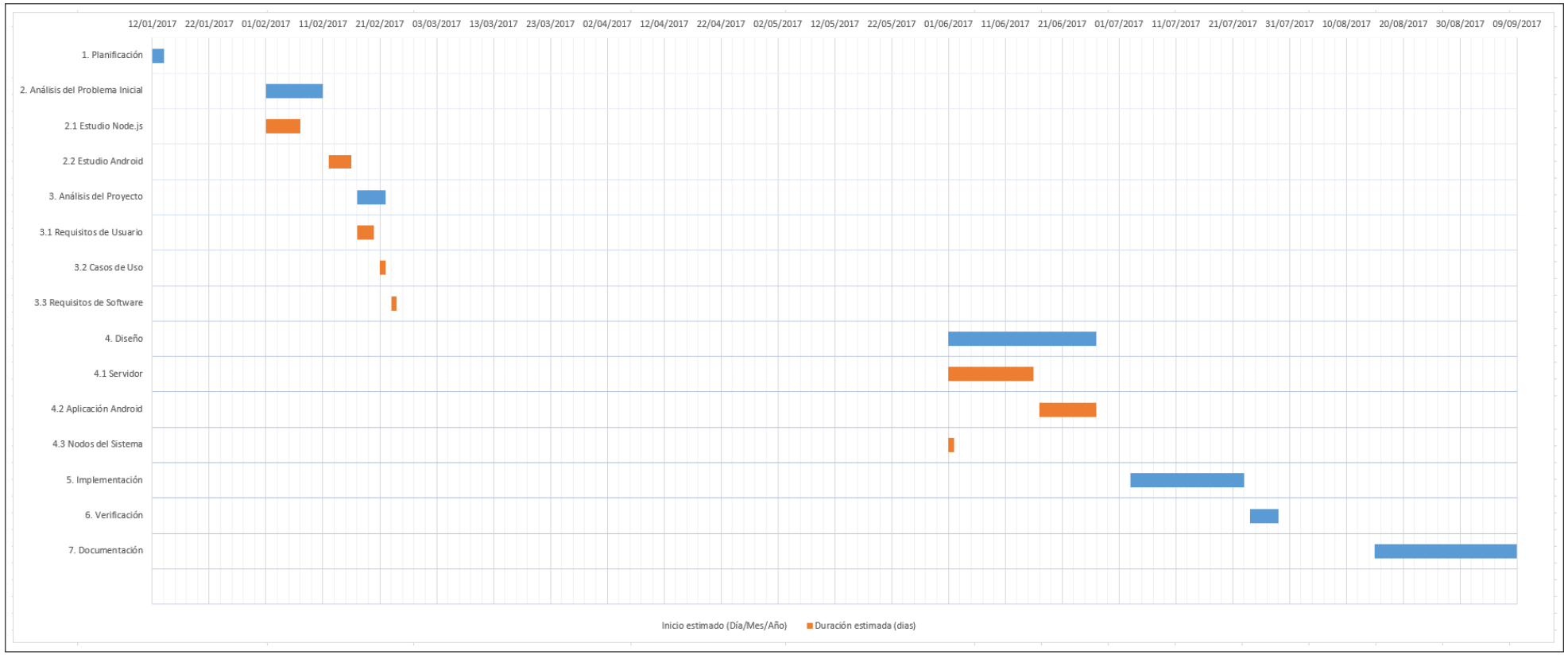

Tabla 7.2: Diagrama de Gantt Planificación Final

A continuación se listan los sucesos y dificultades que provocaron las modificaciones en la planificación inicial:

- El principal motivo que causó esta gran demora es la fuerte carga de trabajos y prácticas de las asginaturas del segundo cuatrimestre. En el momento que se hizo la planificación inicial no se tuvo en cuenta esta gran sobrecarga porque todavía no se habían realizado las presentaciones de las asignaturas del segundo cuatrimestre y no se podía conocer la verdadera magnitud de trabajo que iban a requerir. Esta carga de trabajo incrementó aún más debido a ciertos problemas con los compañeros de grupo.

- La segunda causa fueron los exámenes parciales, tanto teóricos como de prácticas. A pesar de que el segundo cuatrimestre del último curso tiene un número menor de asignaturas, la cantidad de exámenes de cada una ellas fue bastante grande, lo que tampoco se pudo tener en cuenta en la planificación inicial.

- Pasado el período de evaluación continua y el corto espacio de tiempo antes de los exámenes finales, no se pudo continuar el proyecto hasta la culminación de todos los exámenes.

Las horas totales dedicadas a este trabajo se estiman en la siguiente tabla, en la cual se asignan 3 horas de trabajo a cada jornada. El resultado final es de 348 horas de trabajo. Figura 7.1.3.

Etapas del proyecto	Duración estimada (días)	Inicio estimado (Día/Mes/Año)	Fin estimado	Horas estimadas (3 horas por día)
1. Planificación	**2**	**12/01/2017**	**14/01/2017**	**6**
2. Análisis del Problema Inicial	**10**	**01/02/2017**	**11/02/2017**	**30**
2.1 Estudio Node.js	6	01/02/2017	07/02/2017	18
2.2 Estudio Android	4	12/02/2017	16/02/2017	12
3. Análisis del Proyecto	**5**	**17/02/2017**	**22/02/2017**	**15**
3.1 Requisitos de Usuario	3	17/02/2017	20/02/2017	9
3.2 Casos de Uso	1	21/02/2017	22/02/2017	3
3.3 Requisitos de Software	1	23/02/2017	24/02/2017	3
4. Diseño	**26**	**01/06/2017**	**27/06/2017**	**78**
4.1 Servidor	15	01/06/2017	16/06/2017	45
4.2 Aplicación Android	10	17/06/2017	27/06/2017	30
4.3 Nodos del Sistema	1	01/06/2017	02/06/2017	3
5. Implementación	**20**	**03/07/2017**	**23/07/2017**	**60**
6. Verificación	**5**	**24/07/2017**	**29/07/2017**	**15**
7. Documentación	**25**	**15/08/2017**	**09/09/2017**	**75**
	134 días			**279 horas**

Tabla 7.3: Horas totales de trabajo

7.2. Presupuesto

El coste total de realizar este proyecto se expone a continuación. Dentro de este presupuesto se incluye el coste relativo al personal contratado, el coste de los materiales utilizados (físicos y no físicos) y costes indirectos.

7.2.1. Coste de Personal

El proyecto ha sido realizado en su totalidad por el autor del trabajo, sin embargo durante su realización ha ejercido un abanico de roles o cargos que en un proyecto real ocuparían diversas personas. Dentro de estos roles podemos encontrar cargos como programador, jefe de proyecto y responsable de calidad y pruebas. Los salarios asociados a estos empleos durante el período de realización del proyecto se ilustran en la siguiente tabla, la figura 7.4.

Empleo	Salario estimado (anual bruto)	Salario por hora (1080 horas facturables/año)	Horas totales	Salario bruto total	Seguridad Social (6.4%)	IRPF (2%)	Salario neto total
Jefe de Proyecto	34.990,00 €	32,40 €	96	3.110,22 €	199,05 €	62,20 €	**2.848,96 €**
Programador	17.114,00 €	15,85 €	168	2.662,18 €	170,38 €	53,24 €	**2.438,55 €**
Responsable de calidad y pruebas	21.708,00 €	20,10 €	15	301,50 €	19,30 €	6,03 €	**276,17 €**
				6.073,90 €			**5.5563,69 €**

Tabla 7.4: Salarios de empleados

7.2.2. Coste de Materiales

Dentro del cálculo de costes materiales del proyecto se tienen en cuenta los equipos y dispositivos físicos utilizados, así como los productos y licencias software utilizadas. El coste total de estos materiales no va a ser la suma de sus precios, sino que se utilizará una tasa de amortización que determina el coste de emplear dicho material durante el período del proyecto. Esta fórmula se expone a continuación en la figura 7.2.2:

$$\frac{A}{B} * C * D$$

A : Nº de meses de utilización

B : Período de depreciación (60 meses)

C : Coste del componente (sin IVA)

D : Grado de utilización (valor constante 1)

7.2.2.1. Materiales Tangibles

A continuación se calcula el coste de los materiales físicos empleados. Figura 7.5.

Concepto	Coste unitario (sin IVA)	Unidades	Coste total	Meses de uso	Período de depreciación	Tasa de amortización
Ordenador de sobremesa	1.000,00 €	1	1.000,00 €	5	60	**83,33 €**
Samsung Galaxy S4	265,00 €	1	265,00 €	5	60	**22,08 €**
Raspberry Pi 3	35,00 €	1	35,00 €	4	60	**2,33 €**
Teclado y ratón genérico para Raspberry Pi	10,00 €	1	10,00 €	4	60	**0,67 €**
Monitor genérico para Raspberry Pi	85,00 €	1	85,00 €	4	60	**5,67 €**
NodeMCU ESP8266	6,00 €	2	12,00 €	4	60	**0,80 €**
Sensor DHT11	2,00 €	2	4,00 €	4	60	**0,27 €**
						115,15 €

Tabla 7.5: Coste de materiales físicos

7.2.2.2. Materiales No Tangibles

El coste de las licencias del proyecto se representa en la figura 7.6.

Concepto	Coste unitario (sin IVA)	Unidades	Coste total	Meses de uso	Período de depreciación	Tasa de amortización
Microsoft Office 365	42,80 €	1	42,80 €	4	60	2,85 €
						2,85 €

Tabla 7.6: Coste de licencias

7.2.3. Costes Indirectos

Como costes indirectos se incluyen dos meses de abono de transporte que ha sido empleado para los desplazamientos hasta el despacho del tutor. Figura 7.7.

Concepto	Coste unitario	Meses	Total
Abono de Transporte	20,00 €	2	**40,00 €**

Tabla 7.7: Costes indirectos

7.2.4. Coste Total del Proyecto

Uniendo todos estos costes anteriormente calculados se obtiene el coste total de realizar este proyecto. Antes de hacer la suma total se ha de aplicar el IVA a los costes de los materiales empleados. Figura 7.8.

Concepto	Coste	IVA	Coste Total
Materiales Tangibles	115,15 €	21%	**139,33 €**
Materiales No Tangibles	2,85 €	21%	**3,45 €**
			142,78 €

Tabla 7.8: Costes materiales

Finalmente, tenemos como coste total del proyecto el que aparece en la figura 7.9.

Concepto	Coste Total
Personal	5.563,69 €
Material	142,78 €
Indirecto	40,00 €
Total	**5.746,47 €**

Tabla 7.9: Coste total del proyecto

Capítulo 8

Marco Regulador

A lo largo de esta sección se va a explicar el contexto legal en el que se mueve el proyecto. Va a constar de tres apartados: legislación aplicable sobre la implementación, estándares y licencias empleadas y propiedad intelectual del trabajo.

8.1. Legislación Aplicable sobre la Implementación

Debido a que los datos que se manejan en este proyecto no son de carácter personal, no se aplica ninguna ley de protección de datos.

8.2. Licencias y Tecnologías

Referente a los programas y tecnolgías empleados en el desarrollo de este proyecto, se citan sus licencias de uso:

- **Microsoft Windows:** la licencia de este sistema operativo es de tipo estudiante y fue aportada por la Universidad Carlos III de Madrid.
- **Microsoft Office 365:** el paquete de ofimática empleado para la documentación del proyecto fue utilizado como licencia de estudiante cedido por la Universidad Carlos III de Madrid.
- **ShareLatex:** entorno en el que se desarrolló la memoria del proyecto. Licencia gratuita.
- **Android Studio:** entorno de desarrollo de aplicaciones Android con licencia gratuita.
- **Arduino IDE:** entorno de desarrollo de programas Arduino con licencia gratuita.
- **Ubuntu Mate:** la licencia de este sistema operativo instalado en la Raspberry Pi es gratuita.

- **Node.js:** lenguaje de licencia libre.

8.3. Propiedad Intelectual

Este trabajo ha sido enfocado como una exploración de diversas tecnologías para la creación de un sistema completo. Por tanto, al haber empleado diferentes conceptos, su distribución es libre siempre y cuando se referencie al presente trabajo y autor del mismo.

Capítulo 9

Conclusión y Trabajos Futuros

Este capítulo se centra en dedicado a dar mi opinión personal con respecto al trabajo realizado y a las futuras líneas de trabajo posible que este proyecto deja abiertas.

9.1. Conclusión Final

Respecto al proyecto en sí, me ha parecido un trabajo muy exigente y que requería un alto nivel de dedicación pero que me ha dejado con una satisfacción plena. Este trabajo en un principio no tenía unas líneas u objetivos bien delimitados, pues surgió como una idea nueva que fue propuesta al tutor y poco a poco fue perfilándose en lo que se ha convertido finalmente. Por no tener un guía muy bien definida desde un primer momento nos hemos encontrado por el camino con una serie de adversidades y problemas que, en ocasiones, nos han provocado un quebradero de cabeza. Por tanto, me siento muy orgulloso del resultado final que hemos sido capaces de obtener.

Este proyecto se centra en aplicar una serie de conceptos de computación ubicua y de IoT para mejorar el acceso de los usuarios a la información de un sistema. Pienso que este proyecto se adapta perfectamente a la época actual que estamos viviendo, en la que existe una tendencia a digitalizar prácticamente todas las cosas que nos rodean. Este movimiento tiene como fin facilitar la vida de la gente ofreciendo un acceso sencillo y rápido a la información. Un claro ejemplo de esto podría ser el gran impacto que han tenido las redes sociales como Facebook en la vida de la gente. El mero hecho de poder compartir una serie de fotos o comunicarte a través de mensajes con tu familia o amigos ofrece una sencilla alternativa a la tradicional llamada de teléfono o a quedar en un sitio determinado. Es una realidad que esta simple alternativa se está convirtiendo en el método preferido para compartir información de la gente. Es por ello que pienso que el trabajo realizado, que aprovecha parte de estos conceptos, se adapta tan bien en la sociedad actual.

El trabajo ha consistido en crear un sistema de información distribuido y ubicuo. Este sistema es creado para facilitar el acceso de los usuarios a la información contenida en un entorno, ofreciendo múltiples posibilidades de acceso al mismo. Por un lado se ofrece un acceso vía navegador de un ordenador, en el cual se puede ver una recopilación histórica de datos y al mismo tiempo el valor real de los mismos. La segunda opción es una aplicación para dispositivos Android que también permite obtener una recopilación histórica de los datos. Finalmente, la última forma de acceso se da a través de un navegador de un dispositivo móvil que esté conectado a la red de los nodos del sistema. Pienso que las formas de acceso brindadas pueden ser utilizadas por la mayoría de los usuarios, por lo que me parece una solución viable al proeblam de acceso a la información.

El desarrollo del proyecto ha sido bastante tedioso debido a que se ha tenido que lidiar con muchos dispositivos y tecnologías diferentes, lo que han provocado un gran retraso en el aprendizaje de los mismos. Se han tenido que manejar más de cinco lenguajes diferentes, cada uno con ciertas librerías específicas. De la misma forma, el servidor construido para la Raspberry Pi ha sido muy complejo de desarrollar por el hecho de cambiar continuamente de red pero teniendo que mantener la funcionalidad de atender peticiones de los clientes al mismo tiempo.

Sin emabargo, desarrollar este trabajo me ha aportado una gran cantidad de conocimientos de diversas áreas en las que no había profundizado tanto. Sin ir más lejos, el hecho de resolver el problema de comunicación entre el servidor y la aplicación Android con un datagrama UDP era algo que no imaginaba en un principio, pero que a base de investigar mucho vi que era una posible solución al problema. Esta tecnología, los datagramas UDP en broadcast, era algo que conocía únicamente de manera teórica y que nunca pensé que llegaría a utlizarlo en una aplicación. Como esto, he aprendido a manejar muy bien muchas otras tecnologías que creo que me pueden ser de mucha utilidad en un futuro profesional, pues he tocado temas de nivel bajo como pueda ser el desarrollo de un servidor hasta hacer una página web o la apliación Android.

La planificación del proyecto no ha sido la más adecuada en mi opinión, pero para todos incovenientes y retrasos que han surgido en el proyecto creo que se ha llevado de la mejor manera posible. En futuros proyectos tendré más en cuenta la curva de aprendizaje de las tecnologías que se vayan a emplear.

9.2. Trabajos Futuros

El trabajo está enfocado en implementar un sistema que actúa como estación meteorológica que recoge una serie de parámetros del ambiente y que pueden ser leídos por los usuarios. Sin embargo, esta temática puede ser sustituida por otra cualquiera de una manera sencilla. Basta con cambiar los sensores asociados a los nodos para tener una temática diferente. Alguno de estos nodos podría llevar un actuador que pudiera ser activado por los usuarios o el propio sistema al detectar un rango de valores en un parámetro, de tal forma que se pudiera intercatuar con elementos del entorno aparte de leer sus valores.

Uno de los añadidos que fue planteado para este proyecto pero que debido a la falta de tiempo no se pudo llevar a cabo es el de hacer un reconocimiento de voz para el sistema. A través de la aplicación Android se podrían enviar comandos de voz al servidor para que fueran interpretados y ejecutados. Por ejemplo, un usuario podría pedir el valor en tiempo en real de un sensor y que el servidor se lo devolviera, o podría activar una bomba de agua que regaría una planta. Incluso podría enviar un comando que pusiera al servidor a monitorizar la humedad de agua y cuando detectara que el rango de humedad no era el adecuado, el servidor la regaría automáticamente.

Capítulo 10

Apéndice

10.1. Manual de Usuario

Se va a hacer una breve explicación de cómo manejar las distintas aplicaciones que ofrece el sistema, incluyendo su propia instalación.

Los ficheros se encuentran alojados en una cuenta de GitHub y el código fuente de la aplicación Android en una cuenta de Google Drive. Los ficheros de GitHub se pueden descargar desde la siguiente dirección: https://github.com/Marioiottfg/TFG.

- **Arduino:** en esta carpeta encontramos los dos ficheros que contienen el código de los nodos del sistema y una carpeta con librerías necesarias para su ejecución.
- **myApp:** este fichero compone el servidor de la Raspberry Pi.
- **app-release.apk:** este fichero es la aplicación Android compilada lista para ser instalada en cualquier dispositivo Android.

El código fuente de la aplicación Android no pudo ser subido a GitHub debido a su gran tamaño, por lo que se ofrece un enlace a Google Drive desde el cual se puede descargar el archivo comprimido: https://drive.google.com/file/d/0B5XycoSe41BwNkdZcGk3Z3F6MlU/view?usp=sharing.

10.1.1. Raspberry Pi

La aplicación de la Raspberry Pi se ha desarrollado en una Raspberry Pi 3 Model B, con sistema operativo Ubuntu Mate.

10.1.1.1. Instalación

Antes de instalar la aplicación en la Raspbery Pi, se deberá hacer la instalación de una serie de componentes.

El primer componente a instalar es Node.js, cuya versión para este proyecto es 4.2.6. Para ello, en la terminal de comandos del sistema se introducen los siguientes comandos:

```
$ sudo apt-get update
$ sudo wget http://nodejs.org/dist/v4.2.6/node-v4.2.6-linux-armv6l.tar.gz
$ tar -xvf node-v4.2.6-linux-armv6l.tar.gz
$ cd node-v4.2.6-linux-armv6l
$ sudo cp -R * /usr/local/
```

El siguiente paso es actualizar el gestor de paquetes npm, que viene por defecto con Node.js. En la misma carpeta raíz del proyecto introduciremos el siguiente comando:

```
$ sudo apt-get install npm
```

A continuación, utilizaremos este gestor de paquetes para instalar todas las dependencias del proyecto. En la misma carpeta raíz introducimos el comando:

```
$ npm install
```

El navegador recomendado para utilizar la aplicación es Chromium, que se puede instalar de la siguiente forma:

```
$ sudo apt install chromium-browser
```

10.1.1.2. Uso de la Aplicación

Una vez completado correctamente el proceso de instalación de la aplicación se puede hacer uso de la misma. El primer paso será navegar hasta la carpeta raíz del proyecto (*myApp*), en el cual se ejecuta el siguiente comando:

```
$ npm start
```

Acto seguido, se abre un navegador web y se introduce la siguiente dirección: *http://localhost:3000*. Debería aparecer una pamtalla parecida a 10.1:

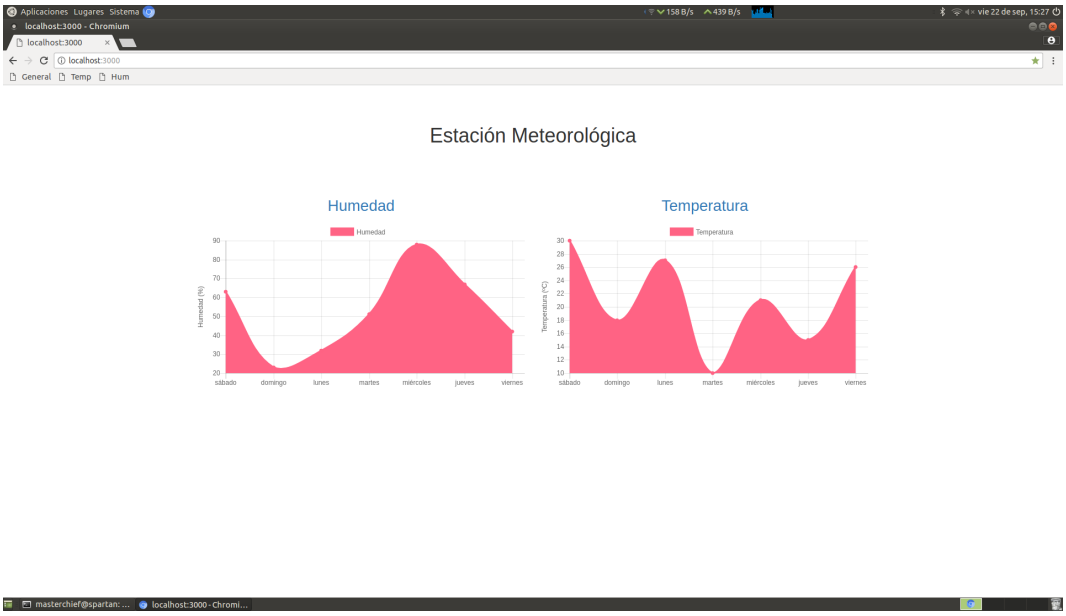

Figura 10.1: Uso de la Aplicación Interfaz Web Pantalla Principal

Si se pulsa uno de los títulos de cada nodo que aparecen en la parte superior de las gráficas o se introduce la url de los mismos (*http://localhost:3000/Temperatura* o *http://localhost:3000/Humedad*) se abre una nueva página con la información de ese nodo. Figure 10.2:

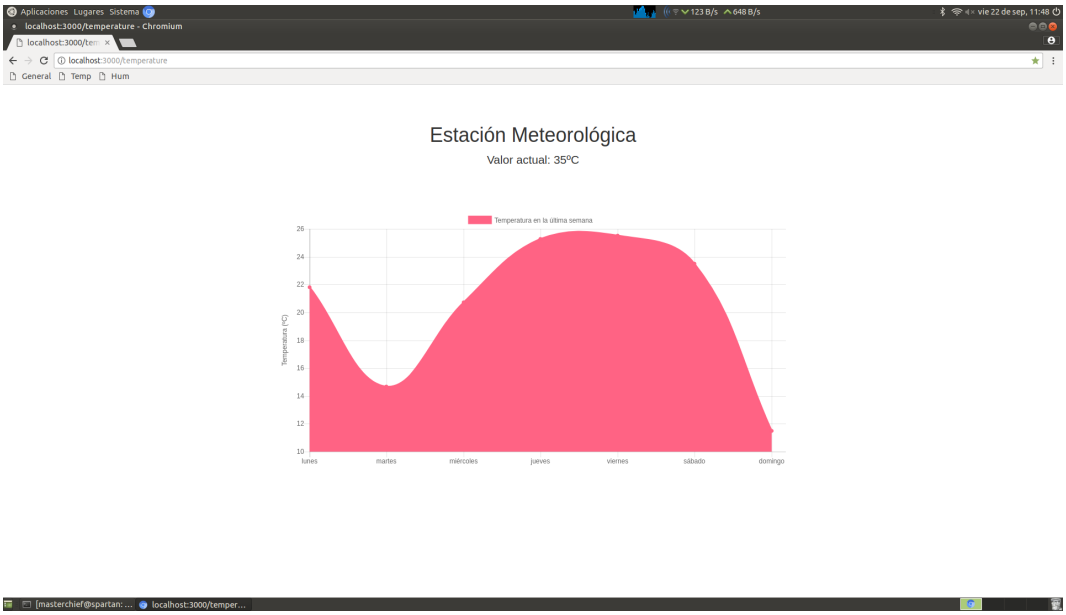

Figura 10.2: Uso de la Aplicación Interfaz Web Pantalla Individual

10.1.2. Aplicación Android

La aplicación Android, denomida Jarvis, ha sido desarrollada para dispositivos de esta plataforma. Todas las pruebas de esta aplicación han sido llevadas a cabo en un Samsung Galaxy S4 físico, no emulado, con Android 7.1.2.

10.1.2.1. Instalación

La instalación de la aplicación es muy sencilla, simplemente basta con descargar el fichero apk y copiarlo dentro del dispositivo móvil. Bastará con buscar ese fichero dentro del dispositivo y hacer click sobre él.

La opción orígenes desconocidos debe estar activada para instalar la aplicación. En caso de no estar activada, navegar a: Ajustes → Seguridad → Orígenes desconocidos.

10.1.2.2. Uso de la Aplicación

Antes de iniciar la aplicación es recomendable activar el wifi, en caso de que no lo estuviera, y desactivar los datos móviles.

Tras hacer click en la aplicación aparece una pantalla de carga. Esta pantalla tarda unos cuantos segundos en cargar pues está interpretando los mensajes de broadcast del servidor y esperando la respuesta del mismo. Suele tardar unos 20-30 segundos. Figura 10.3:

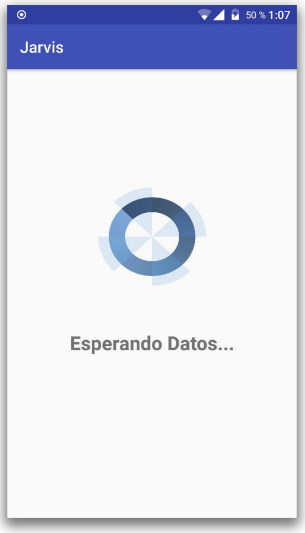

Figura 10.3: Uso de la Aplicación Android Pantalla 1

Una vez se hayan recibido los datos correctamente, se cambiará la pantalla para mostrar la red del entorno con los elementos que la componen, que son imágenes sobre las que se puede hacer click para acceder a su información. Figura 10.4:

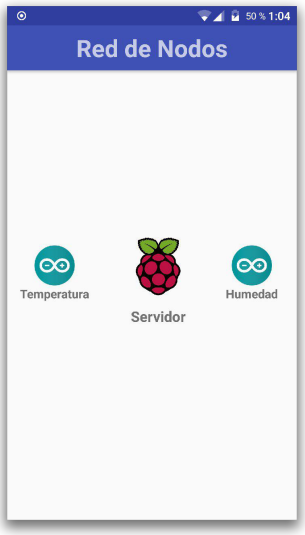

Figura 10.4: Uso de la Aplicación Android Pantalla 2

Si se pulsa sobre el botón del servidor, que está definido con el logo de Raspberry Pi, aparece la información de todos los nodos del sistema. Figura 10.5:

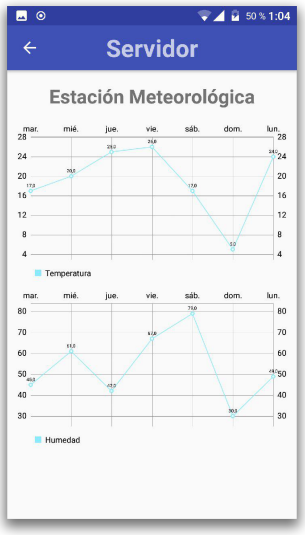

Figura 10.5: Uso de la Aplicación Android Pantalla 3

Si se pulsa cualquiera de las otras imágenes, se abre una pantalla con la información de ese nodo en concreto. Figura 10.6:

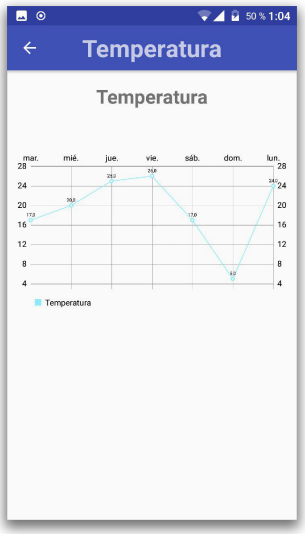

Figura 10.6: Uso de la Aplicación Android Pantalla 4

Se puede navegar hasta el menú principal si se pulsa el botón de atrás del dispositivo móvil o la flecha que aparece en la esquina superior izquierda.

10.1.3. Nodos del Sistema

Los nodos del sistema estarán representados por el microcontrolador NodeMCU ESP8266 y el sensor DHT11. A continuación se explica el proceso de instalación y manejo de la aplicación desarrollada para los nodos del sistema.

10.1.3.1. Instalación

La instalación de los nodos del sistema se llevará a cabo en dos pasos: conectar el microcontrolador al sensor y cargar el programa de Arduino en el microcontrolador.

Para realizar el primer paso, conectar el sensor al microcontrolador, se necesitarán los siguientes elementos:

1. Cables de puente macho a hembra.

2. Protoboard (opcional).

3. Sensor DHT11.

4. Resistencia 10 kΩ.

5. NodeMCU ESP8266.

6. Cable USB a Micro USB.

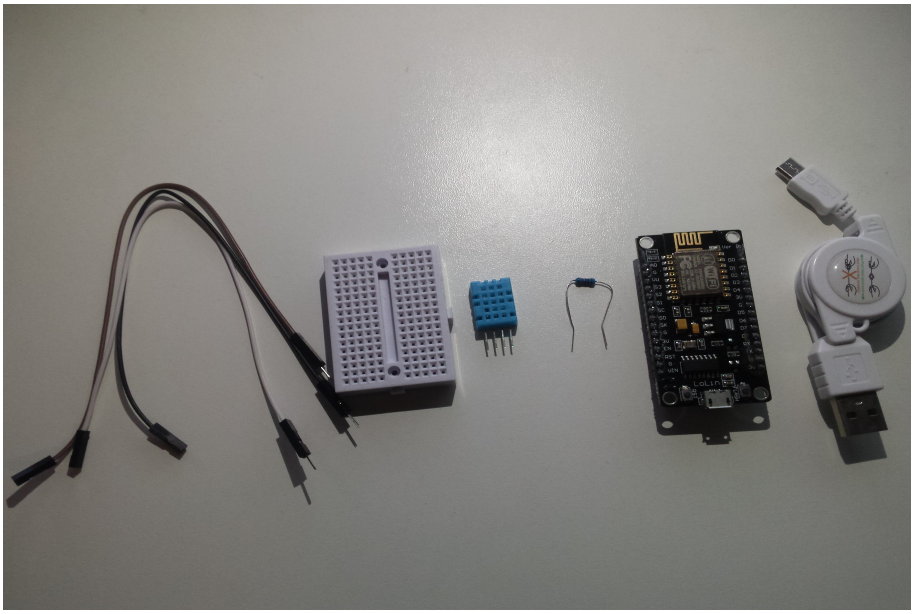

Figura 10.7: Instalación NodeMCU Componentes

El primer paso será conectar el sensor a la protoboard, para lo cual insertamos las patas en los agujeros presionando levemente. El sensor tiene dos caras, una es lisa y la otra tiene agujeros. Las siguientes conexiones se harán tomando como referencia la cara con agujeros del sensor. Figura 10.8.

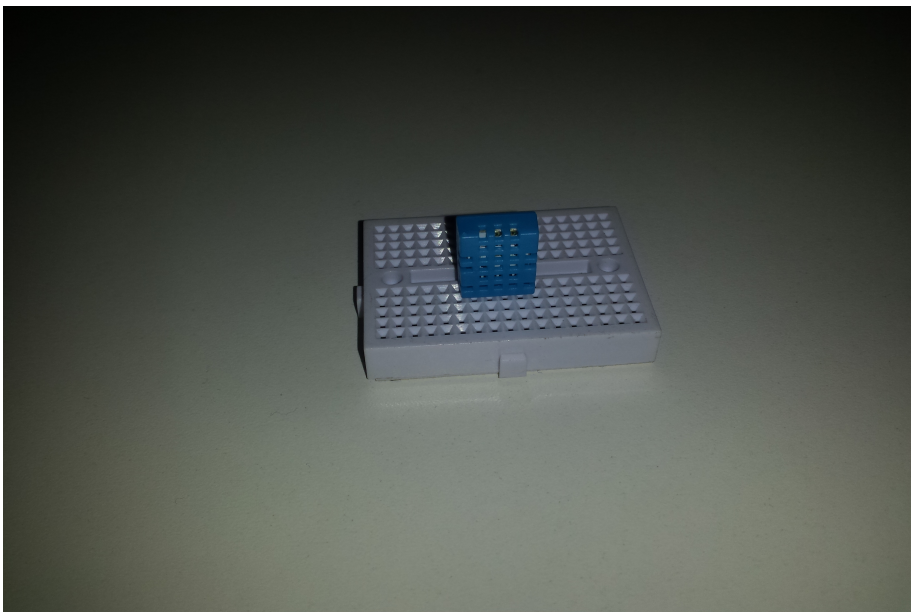

Figura 10.8: Instalación NodeMCU Paso 1

Posteriormente, se conecta la resistencia a las dos primeras patas del sensor empezando por la izquierda. Figura 10.9

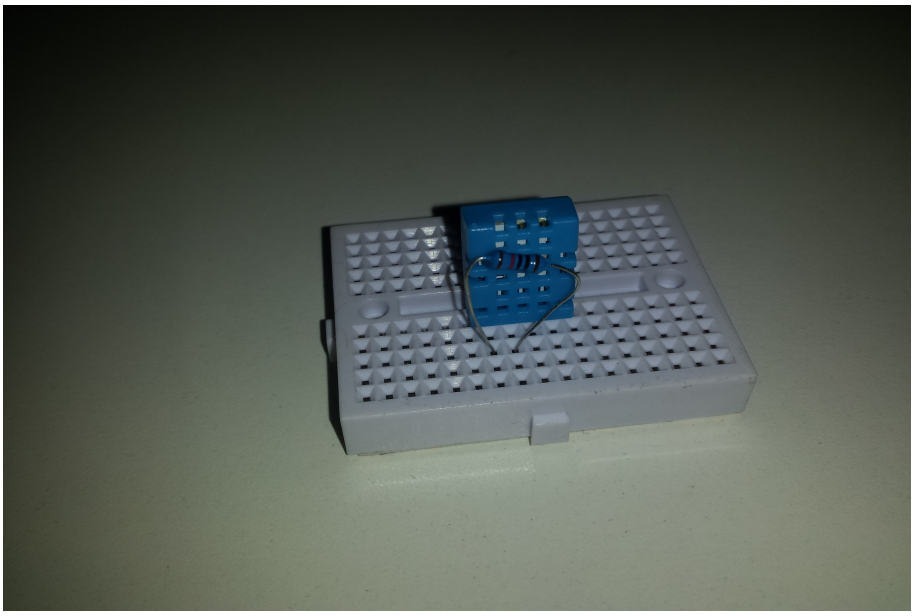

Figura 10.9: Instalación NodeMCU Paso 2

Seguidamente, se conectan tres cables de puente a las dos primeras patas empezando por la izquierda y a la última de ellas, dejando la tercera sin conectar a nada. El cable de más a la izquierda se conecta a la toma de voltaje 3v3. El segundo cable por la izquierda se conecta al pin digital D5. El último cable se conecta a la toma de tierra o ground de la placa. Figura 10.10

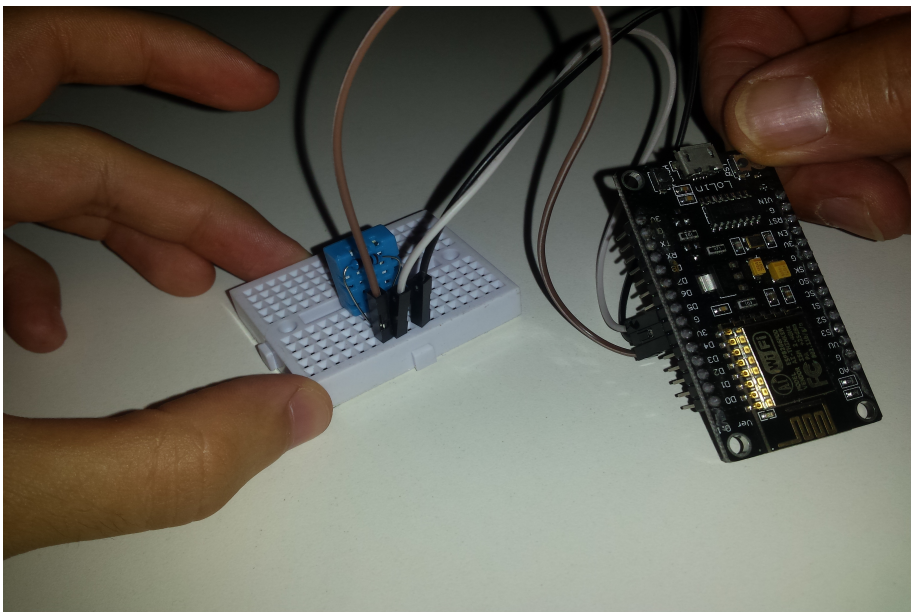

Figura 10.10: Instalación NodeMCU Paso 3

Antes de proceder a instalar el código en el microcontrolador se deberán hacer unas instalaciones previas. El primer componente a instalar es el driver CH341SER del microcontrolador, que se puede encontrar en el siguiente enlace: http://www.wch.cn/download/CH341SER_EXE.html

El siguiente componente a instalar, si no estuviera, es el entorno de desarrollo Arduino IDE, que se puede descargar del siguiente enlace: https://www.arduino.cc/en/Main/Software.

Una vez abierto el programa, se deberá incluir el repositorio para la instalación del chip ESP8266 de la placa microcontroladora. Para ello, navegaremos a: Archivo (Windows) o Arduino (Mac OS X) → Preferencias.

En la parte inferior de esta ventana encontraremos una entrada que se llama: Gestor de URLs Adicionales de Tarjetas. Introduciremos en la entrada el siguiente enlace: http://arduino.esp8266.com/stable/package_esp8266com_index.json. Figura 10.11.

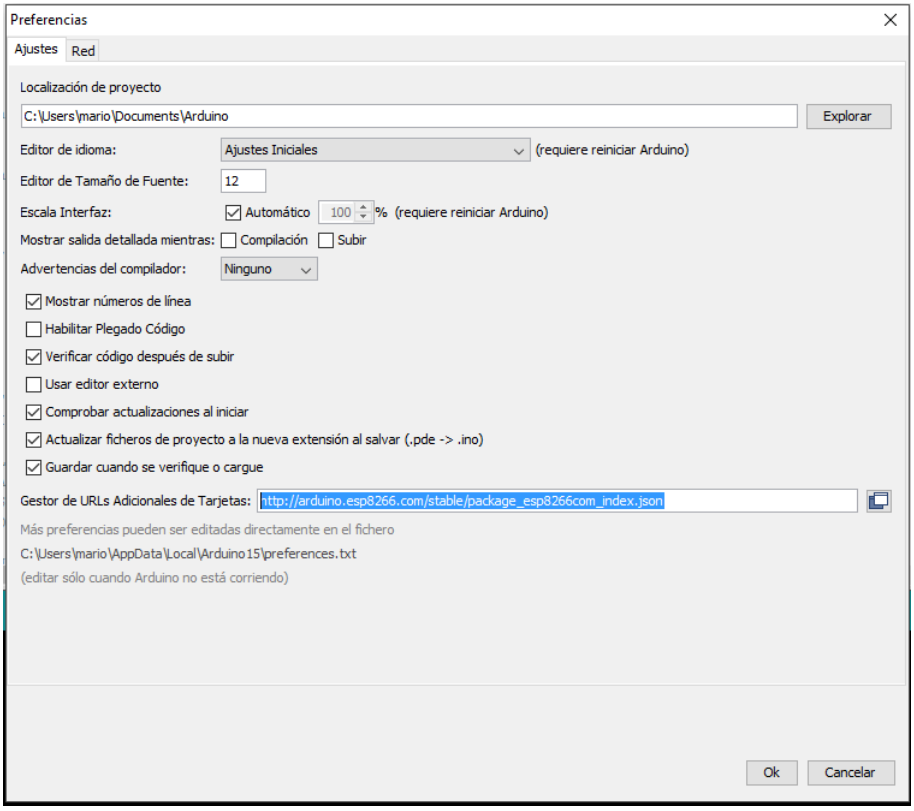

Figura 10.11: Instalación NodeMCU Paso 4

Después, navegaremos en la barra del menú superior de Arduino a: Herramientas → Placa:'...' → Gestor de Tarjetas... Introduciremos en la barra de búsqueda: *esp8266*. Pulsaremos el botón de instalar y cuando se haya completado la operación, en la lista de placas seleccionaremos: NodeMCU v1.0 (ESP-12E).

Finalmente, la instalación del código en la placa se realizará pulsando la flecha que apunta a la derecha en la esquina superior izquierda o pulsando el comando Ctrl + U. En la ventana de comandos negra de la parte inferior deberá aparecer el éxito de la subida.

10.1.3.2. Uso de la Aplicación

Una vez ha sido cargada la aplicación en el microcontrolador, la placa deberá estar conectada mediante el cable USB a una fuente de alimentación (por ejemplo: un ordenador). Tras esto, el programa habrá empezado a ejecutarse y se habrá creado el servidor web.

Para acceder al servidor, se utilizará cualquier dispositivo portátil que cuente con wifi (por ejemplo: un móvil) y se conectará la red wifi al punto de acceso cuyo nombre empiece por el prefijo *"mgiot"*. Es recomendable que el dispositivo se desconecte de la red de datos móviles y sólo tenga acceso a la conexión wifi, porque ésta al no ofrecer conexión a Internet puede hacer que el dispositivo no se conecte.

Seguidamente, se abre el navegador web del dispositivo y se introduce la url: *192.168.4.1*. Se deberá ver una pantalla del estilo de la figura 10.12:

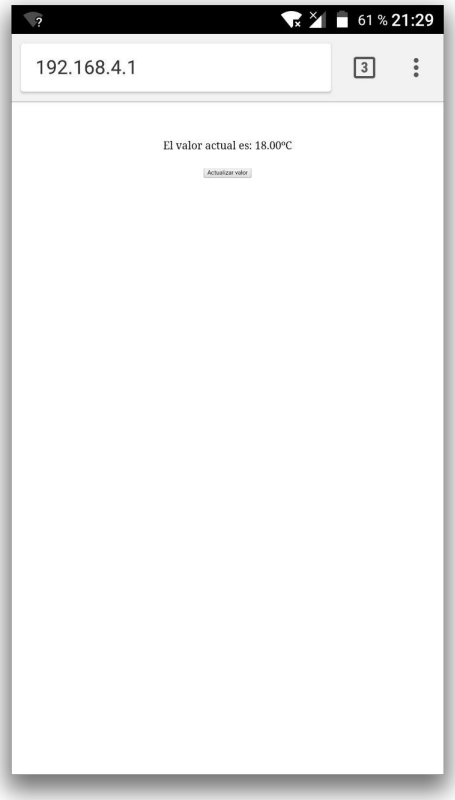

Figura 10.12: Uso de Aplicación de Arduino

10.2. Fallos encontrados

En el sistema se pueden dar una serie de errores o bugs que se listan a continuación.

10.2.1. Raspberry Pi

- Durante la ejecución del programa del servidor, principalmente cuando se está accediendo a la interfaz web, se puede producir un fallo debido a un *"socket hang up"*. Esto puede producirse porque los nodos no respondan a la petición del servidor en un tiempo determinado o porque llegue un nuevo evento al servidor mientras esté resolviendo una tarea. La principal causa es la asincronía de la ejecución.

- Al abrir la página principal o alguna de las específicas no carga la gráfica que muestra los datos. A veces, en la página principal donde se muestran dos gráficas sólo aparece una.

10.2.2. Aplicación Android

- La aplicación no pasa de la pantalla de carga. A veces, el servicio de escucha no recepciona el mensaje enviado por el servidor o las peticiones enviadas de la aplicación al servidor no son respondidas. La mejor solución es probar a reiniciar la aplicación.

10.2.3. Nodos del sistema

- Valor devuelto por los sensores es cero. Debido a la baja calidad de los sensores y la placa microcontroladora, el valor de temperatura o humedad leído puede ser mal interpretado y asignado como cero. Mejorar la calidad de estos componentes solucionaría el problema pero aumentaría el coste total del proyecto.

Capítulo 11

Summary

This section is an overall summary of the project developed described in English, as it is required for the document to be valid. The structure of this chapter is composed of introduction, objectives, results and personal conclusion.

11.1. Introduction

This project covers the development of an Internet of Things environment, with the thematic of a weather station. Basically, the aim of the environment is to gather weather parameters so that users can read them. The main key in this project is that users can access this information in three different ways. This possibility allows a greater number of users to have access to the system and the information that it contains.

The goal of this project is to develop a system based in ubiquitous computing and Internet of Things concepts. Both terms describe a network of interconnected devices for the purpose of exchanging and retrieving data. Unlike desktop computing, this kind of systems can occur with any device, at any time, in any place and in any data format.

The devices in this environments are usually different from the traditional desktop computers. The most frequent elements in these systems are: computers, microcontrollers, sensors and actuators. Among the most used computers are smartphones and tablets. This is caused by the massive technological development of portable devices. Microcontrollers are a type of computer with limited tasks, and they are usually attached to sensors or actuators to interact with the environment.

11.1.1. Devices

In the next section, the devices that were used for this project will be described. All of them are focused on working in IoT environments, so they are low-cost and have wireless communication.

11.1.1.1. Raspberry Pi

The first item that is being introduced is the Raspberry Pi. This small computer is used in the project and has the following specifications:

- Raspberry Pi 3 Model B
- Quad Core 1.2GHz ARM 64-bit CPU
- 1GB RAM
- Wireless LAN and Bluetooth
- 40 pin GPIO
- 4 USB ports, HDMI and Micro SD port
- Micro USB power source

This single-board computer is a credit-card sized computer with the capability of executing any task that a modern desktop computer can perform. It is focused on educational purposes such as the teaching the basics of computer science or developing applications in some programming languages, such as C or C++. This computers are very cheap, they cost around 30 euros, and have plenty of accesories available. The Raspberry Pi 3 Model B is shown in the picture below, the figure 11.1:

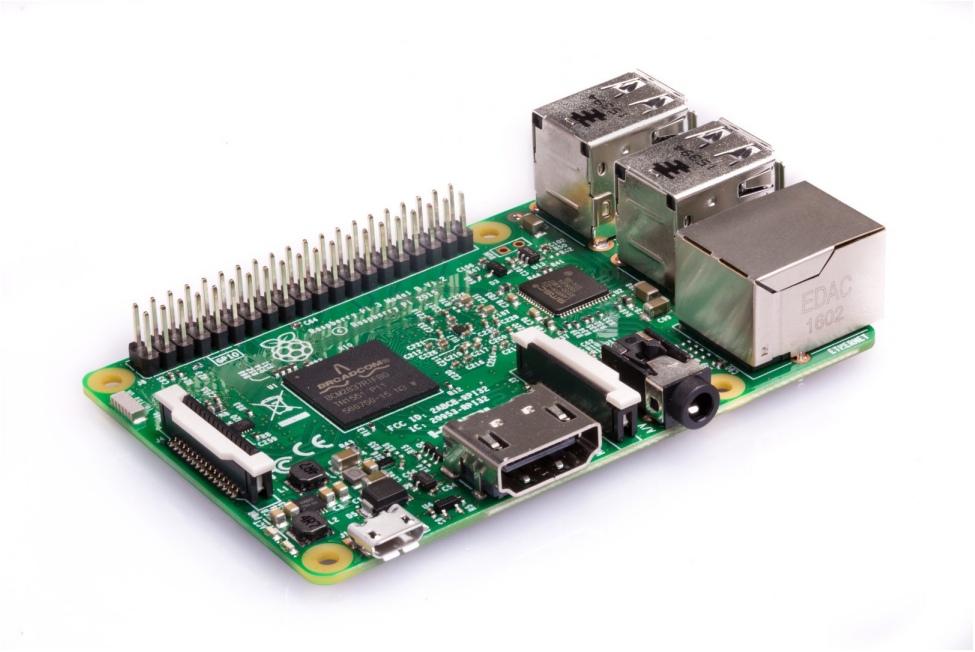

Figura 11.1: Raspberry Pi 3 Model B

The most popular operating system used in this board is Raspbian that is based on a Debian operating system. There are many other operating systems that can be installed, most of them are Linux-based, but some of them are not, like the new Windows 10 IoT Core. In this particular project, the operating system installed is Ubuntu Mate, a version of the original Ubuntu.

Using this computer for IoT environments makes perfect sense for several reasons:

- **Wireless connections:** this device has wifi and bluetooth interface avilable, which are pretty much the standard ways of communicating in this environments.
- **Low price:** IoT environments aim to connect several devices to each other, some of them are everyday objects with an embedded processor, so the lower the price of these devices, the more there will be and more environments can be populated.
- **Small size:** the small proportions of this computer allows it to be located almost anywhere. Regardless, the computational capability is not affected by the size.
- **GPIO pins:** these pins allow the Raspberry Pi to control some sensors or actuators without using any microcontroller as intermediary.

11.1.1.2. NodeMCU

NodeMCU is a single-board microcontroller that includes the ESP8266 WI-FI SoC, which allows this board to use wifi connections. This board is an open-source platform that also focused on educational purposes and DIY projects. The price of this board is very low, around 4 euros. The image below represents this board, figure 11.2

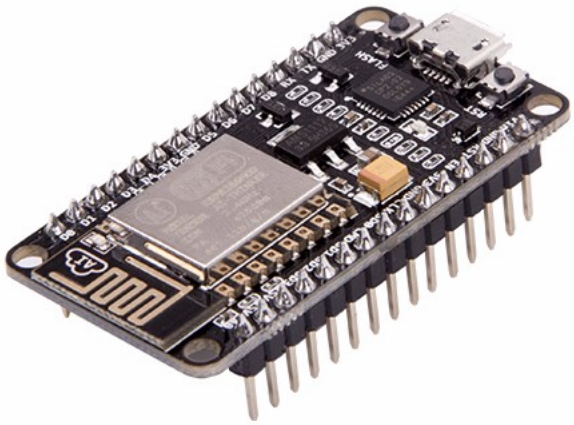

Figura 11.2: NodeMCU

A microcontroller is a small computer that usually acts as an intermediary between a master computer and some sensors or actuators. Most of them are designed to run only one task or program, so they can be embedded in other object or device. While some embedded systems are very sophisticated, many microcontrollers don't even have operating system.

NodeMCU is a very basic microcontroller with a wifi module. This feature allows the board to host a web server while monitoring some sensors or triggering some actuators. The possibilities it offers and the low cost makes the NodeMCU a perfect item to include in an IoT environment.

11.2. Objectives

This work aims to explore the usage of the current Internet of Things board to know what advantages and disadvantages different configurations provide. This work will use the sensor monitoring context to develop a system for three different scenarios.

The first scenario will be a centralized system where the Raspberry Pi acts as a server that hosts a web interface so that users can acces the data. The server will collect the weather parameters by connecting to the access points of the nodes. After the successful connection, the server will send an HTTP request to the node to collect the last value read by the sensor. This data will be stored in a MongoDB database, which is a non-relational database type. The web page will display the historical records of the data gathered from each of the system nodes. Historical records will represent the variance of the data collected by the sensor in the last seven days, and will be displayed in graphs. The page will also display the real-time value of the specific sensor that the user is accesing.

The second option the system offers to the users is the use of a native Android application. This application has the same functionality as the web interface of the server, with the exception that it will not display the real-time value of the nodes. The application will send a request to the server to collect the data stored in the database. Afterwards, a menu will be shown where the server and the current active nodes in the system are displayed as buttons. Clicking one of these buttons will display the information related to that item in a chart. If the server button is clicked, there will be shown as many graphs as there are currently active nodes in the environment.

The last scenario will directly communicate the users to the nodes. To do so, the users will connect their portable devices network to the access point granted by the nodes. Then, they will open the browser and introduce the web address or url of the node, receiving an HTML page with the last value read by the sensor.

The general overview of the system is shown in the picture below, the figure 11.3:

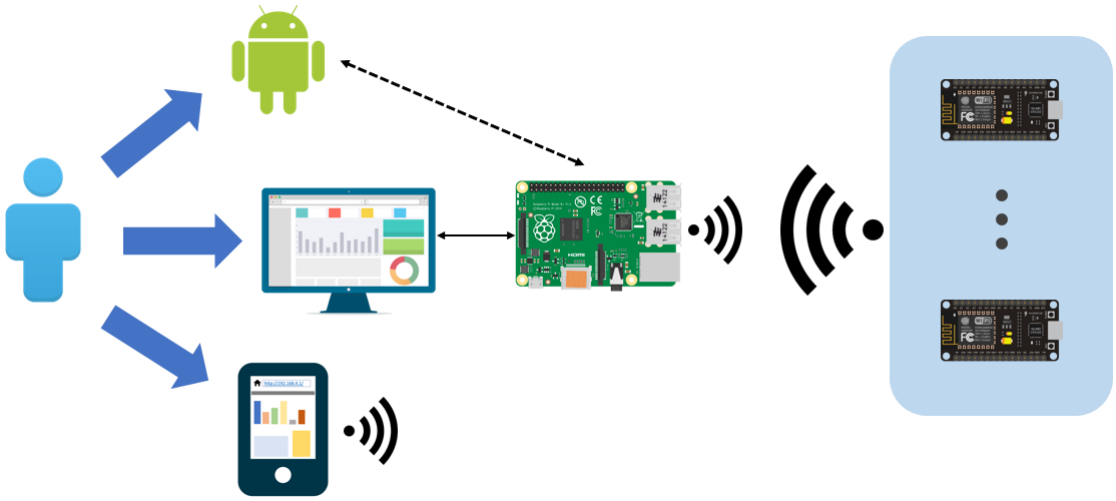

Figura 11.3: General Architecture of the Environment

As shown in the previous picture, the user will have three different ways of accessing the data. The first two scenarios will be centralized systems, with the Raspberry Pi server distributing the information. The last one will allow users to connect directly to the nodes. All three scenarios use wifi as the primary means of communication.

11.3. Results

This chapter describes the system development process for each of its components following the objectives set in the previous section. Three different items are being developed: Raspberry Pi as client-server, Android application and nodes of the system.

11.3.1. Raspberry Pi

The Raspberry Pi board has two roles in the system. One of them is to gather the data collected by the sensors of the nodes. The second one is to return this information to the users, answering incoming requests. These roles make this computer act as a client-server device in the environment.

Ubuntu Mate is the operating system for the Raspberry Pi. The application is developed in Node.js, a Javascript runtime built. The structure of the program is made by the Express.js framework, which creates a very basic but fully functional server application. The web interface hosted by this server is coded in Jade, a template focused on quick HTML coding.

11.3.1.1. Server Application

The main server program is located in the root of the folder created by Express.js, named *app.js*. The functions in charge of answering incoming requests are located in the *routes* folder, which contains several Javascript files.

In order to collect the data from the system nodes, the server connects its wifi to the nodes access points and sends an HTTP request. To do so, the server first scans the available access points in the environment, which share the prefix *"mgiot"* to be identified, stores them in an array and connects to each of them. In the same process of scanning the available wifi networks, the server stores the local wifi network established for the Android applications. This local network allows the Android applications to communicate with server, because the server is constantly changing its access point or network.

Basically, the server code can be divided in two functions: the first one scans the wifi networks available in the area and the second one connects to them. After successfully connecting to the network, there are two possible cases: the connection is being made to a system node, so the server will send a request to get the last value read by the sensor; the second option is the connection to the local wifi network established, where the server sends a broadcast datagram through the network to notify Android applications that a connection to the server can be made.

11.3.1.2. Database

The data gathered from the nodes is stored in a MongoDB database. This is a non-relational database that uses collections instead of tables. Each collection is composed of documents, which are the stored records. The format used in this database is JSON.

The database is implemented in the server application using a middleware called Monk. This library returns a very easy to use database handler, which simplifies the operations with the database.

The first interaction with the database in the *app.js* file is the creation of this handler. After that, the server starts scanning the networks available in the environment, inserting the name of those that belong to the system in a collection with that purpose, if they are not yet registered.

After the scanning process, the server starts connecting to the stored networks. If a connection is made to a system node, the server sends an HTTP request to get the last value read by the sensor. Once this data is available it is stored in a collection named after the node's name. The record is composed of the value collected and the timestamp of the insertion.

There are more operations outside the file, which are very similar to each other. These operations are located in the *routes* files and restore the data stored in the database to answer requests. Two kind of requests can be sent to the server: the first one asking for the historical record of a specific node or a second one asking for the historical record of all the system nodes.

The request for the historical record of a specific node is sent by the individual pages of the web interface. The first step for the server is to get the requested node collection handler. After that, a query is sent to the database with the parameters set to get the records inserted in the last seven days. Once the data is restored from the database, it is loaded in the Jade file to render the web page.

The request for the historical records of all the system nodes is very similar. This request is sent by the Android application and the index page of the web interface. First of all, the server collects all the node names from the names collection. Then, it sends the same query to each collection. After collecting all the data, there are two cases: if the request was sent by the Android application the data is returned in JSON format; if the request was sent by the index page, the data is loaded to the Jade file and the web page is rendered.

11.3.1.3. Address routing

The address routing for the web interface is defined in the *app.js* file. All address routes are linked to the *routes* and *views* files that answer those requests. There is an index page that displays all the system nodes historical records and multiple pages that display each node hisotrical record.

Pages that display a historical record show the variance of the data collected in the last seven days on a chart. They also display the last sensor real-time value, which is the last value recorded by the server.

The index page displays the variance of the data collected in the last seven days from all the nodes in charts, but it does not display the real-time values.

11.3.1.4. Networks Switching

This process has two goals: gathering data from the system nodes and open communication with Android applications. To do this, the server changes its wifi network between the system nodes access points and the local wifi network established.

First of all, the server starts scanning the available networks in the environment. This function is executed in intervals during the excution of the program, so it is executed every 10 seconds. Whenever a network with the prefix *"mgiot"* is found, the server stores it in an array and in a collection in the database, if it was not yet registered. After that, the server stores in the array the local wifi network established and invokes a function that starts the networks switching task with the stored networks in the array.

The second function, in charge of the networks switching, has two objectives: to collect data from the system nodes and to allow Android applications communicate with the server. This function iterates over the networks array, created in the previous function, connecting to each one.

If the network belongs to a system node access point and the connection is successful, the server sends an HTTP request to the node to get the last value read by the sensor. Once this data is available it is stored in the database.

If the network is the local wifi network established and the connection is successful, the server notifies Android applications that the communication is available sending a broadcast datagram. To do this, the server creates an UDP datagram and sends it through the broadcast IP address and the port 11111. Then, the server delays its execution for a few seconds, so that Android applications have time to establish communication with the server before it changes its network.

After connecting to a network, it is removed from the array and the function is invoked with the new array, making this function a recursive type.

11.3.2. Android Application

The application developed implements the same functionality as the Raspberry Pi server web interface, with the exception that it will not display the real-time values.

The main programming language is Java, which is complemented by XML files that represent the design of the application. The application has been developed in the latest version of Android 7.0 Nougat. The structure can be divided in two parts: the first is the reception of data and the second is the representation of it.

11.3.2.1. Data Gathering

At the start of the application, the first thing to do is to restore the system nodes data from the server. The application can not send an HTTP request to the server directly because it does not know its IP address beforehand. So, before sending this request to the server, the application has to get the server IP address.

In order to get the server IP address, the application catches one UDP datagram the server sends, because its IP address is encapsulated inside it. When the application starts, a listener service is created. This service is listening for UDP datagrams, and it stops when a datagram is catched. After getting the server IP address, the application sends an HTTP request to the server, asking for the system nodes historical record. Once the data is collected, the application stores it in the device and navigates to the main menu.

The collected data is stored in the device as SharedPreferences, which is basically a key-value file that can be private and accessed by the application only.

11.3.2.2. Data Representation

Once the data is collected and stored in the system, the application navigates to the main menu where it shows all the system nodes available and the server as pictures. These pictures are buttons that, when they are pressed, navigate the user to a new page that displays that item information.

If a node image-button is tapped, the application displays the data collected at last seven days by the node in a chart.

If the server picture-button is tapped, the application displays the data collected at last seven days by all the system nodes in multiples charts.

The user can return to the main menu by clicking the upper left arrow.

11.3.3. System Nodes

The system nodes gather the environmental parameters and distribute them to users. They are composed of a microcontroller NodeMCU ESP8266 and a sensor DHT11. The microcontroller is open source and includes the necessary firmware to carry a wifi module, which can be set as an access point. The mentioned sensors can read the parameters of temperature and humidity of the environment. To access this parameters by wifi connection, the application hosts a web server.

This application development is made in Arduino IDE, which uses Arduino as programming language. This application also returns a web page, which is coded in HTML.

The application is composed of three functions: the first one is the *setup* of the program and the rest are functions that answer the incoming requests.

The first method, the *setup* function, initializes the web server and links the address routes that the server is going to answer. The web server name is composed by the prefix *"mgiot"* followed by the name of the sensor that it uses.

Incoming requests to the server can be of two types: the first one handles the Raspberry Pi server requests and the second one answers the portable devices browser requests.

Whenever Raspberry Pi server sends a request to the node, the first thing to do is reading the value of the sensor. This value is sent to the server in text format.

If the request is sent from a device browser, the node reads the value of the sensor and loads it in an HTML page, which displays the value of the sensor and a button that reloads the page when it is pressed.

11.4. Personal Conclusion

First of all, I would like to thank all the effort and help that the mentor has given me throughout this project to make it possible. Without this support I could never have finished this work.

As for the project, I am really happy with the results obtained. It has been very tough sometimes and required a high level of knowledge of various subjects. The idea for this project came from a series of meetings with the mentor, but the purpose and the objectives were not well defined. It has evolved into what it is now, a fully functional IoT environment with three different ways of accesing the data.

Bibliografía

[1] Buscando ideas contra la niebla en la a-8. [Online]. Available: https://www.lavozdegalicia.es/noticia/galicia/2015/06/07/buscando-ideas-contra-niebla-a-8/0003_201506G6P10991.htm

[2] Gartner says 8.4 billion connected. [Online]. Available: www.gartner.com/newsroom/id/3598917

[3] M. A. A. Swapna, "Implementation of sensor data monitoring and transmitting using raspberry pi with reference to healthcare industry." [Online]. Available: http://www.ijmetmr.com/oljuly2015/ASwapna-MdAmmeenuddin-324.pdf

[4] A. K. Dennis, "Raspberry pi home automation with arduino," 2013.

[5] G. R. Gonzáluez, "Evaluación de introducción de internet de objetos en espacios de aprendizaje," 2010. [Online]. Available: https://www.researchgate.net/profile/Gustavo_Ramirez-Gonzalez/publication/49215882_Evaluacion_de_introduccion_de_Internet_de_objetos_en_espacios_de_aprendizaje/links/0fcfd50b6423541423000000.pdf

[6] M. N. Kamel Boulos and S. Wheeler, "The emerging web 2.0 social software: an enabling suite of sociable technologies in health and health care education," *Health Information & Libraries Journal*, vol. 24, no. 1, pp. 2–23, 2007.

[7] A. Nordrum. Popular internet of things forecast of 50 billion devices by 2020 is outdated. [Online]. Available: https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

[8] A. K. M. G. Olegs Nikisins, Rihard Fuksis, "Face recognition system on raspberry pi." [Online]. Available: http://bite.edi.lv/wp-content/uploads/2015/04/ICIPCE2015_Nikisins_paper.pdf

[9] G. Pannu, M. Ansari, and P. Gupta, "Design and implementation of autonomous car using raspberry pi," vol. 113, pp. 22–29, 03 2015.

[10] K. N. R. Deepan, Santhana Vikrama Rajavarman, "Hand gesture based control of robotic hand using raspberry pi processor," 2015. [Online]. Available: http://docsdrive.com/pdfs/ansinet/ajsr/2015/392-402.pdf

[11] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[12] M. Weiser, "The computer for the 21st century," 1991. [Online]. Available: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html